Simulatable Commitments and Efficient Concurrent Zero-Knowledge

Daniele Micciancio[1] * and Erez Petrank[2] **

[1] Computer Science and Engineering Dept.,
University of California, San Diego, La Jolla, California.
daniele@cs.ucsd.edu.
[2] Dept. of Computer Science,
Technion - Israel Institute of Technology, Haifa 32000, Israel.
erez@cs.technion.ac.il

Abstract. We define and construct *simulatable commitments*. These are commitment schemes such that there is an efficient interactive proof system to show that a given string c is a legitimate commitment on a given value v, and furthermore, this proof is efficiently simulatable given any proper pair (c, v). Our construction is provably secure based on the Decisional Diffie-Hellman (DDH) assumption.
Using simulatable commitments, we show how to efficiently transform any public coin honest verifier zero knowledge proof system into a proof system that is *concurrent* zero-knowledge with respect to *any* (possibly cheating) verifier via black box simulation. By efficient we mean that our transformation incurs only an *additive* overhead (both in terms of the number of rounds and the computational and communication complexity of each round), and the additive term is close to optimal (for black box simulation): only $\omega(\log n)$ additional rounds, and $\omega(\log n)$ additional public key operations for each round of the original protocol, where n is a security parameter, and $\omega(\log n)$ can be any superlogarithmic function of n independent of the complexity of the original protocol. The transformation preserves (up to negligible additive terms) the soundness and completeness error probabilities, and the new proof system is proved secure based on the DDH assumption, in the standard model of computation, i.e., no random oracles, shared random strings, or public key infrastructure is assumed.

1 Introduction

Zero knowledge proofs are (interactive) proofs that yield nothing but the validity of the assertion being proved, and they are one of the most fundamental building blocks in cryptographic protocols. For example, zero knowledge proofs can be used to make sure that distrustful parties involved in a protocol are

* This research was supported in part by NSF Career Award CCR-0093029.
** This research was supported by the Technion V.P.R. Fund - N. Haar and R. Zinn Research Fund.

really following the protocol instructions, without revealing any extra information. The original formulation of the notion of zero knowledge [22] considers a single prover and a single verifier working in isolation. This formulation is inadequate for real applications where zero knowledge proofs are used as part of complex protocols. In order to use zero knowledge proofs in real applications one needs to make sure that the proof system is zero knowledge not only when executed in isolation, but also when many instances of the proof system are executed asynchronously and concurrently. This strong notion of zero knowledge, first discussed in [15, 13], has been the subject of many recent investigations [14, 27, 13, 12, 9, 4, 32, 26, 5, 7, 1]. For example, in [9, 4], it is shown that if a public key infrastructure (PKI) is in place, then all languages in NP have an efficient (constant round) concurrent zero knowledge proof system. Unfortunately, in the standard model, where no PKI is available, Canetti, Kilian, Petrank and Rosen [5] have shown that no nontrivial language (i.e., no language outside BPP) has constant round black box concurrent zero knowledge proofs. In particular, [5] provides a $o(\log n/\log\log n)$ lower bound on the number of rounds for any such protocol. Interestingly, Richardson and Kilian [31] presented a concurrent black-box zero knowledge interactive proof for all NP, demonstrating the existence of such a proof. The proof required a polynomial (in the security parameter n) number of rounds. Kilian and Petrank [26] have drastically reduced the number of rounds by providing a simulator and an analysis showing that a polylogarithmic number of rounds is sufficient to achieve concurrent zero knowledge. Recently, the analysis of the simulator of [26] has been improved by Prabhakaran, Sahai and Rosen[30] showing that $\omega(\log k)$ many rounds are enough.

Although less efficient than solutions in the PKI model, the solution of [31, 26, 30] is interesting because it may be used where a PKI is not possible, or as a mean to set up a public key infrastructure or establish common random strings. Namely, one possible good use of a moderately efficient concurrent zero-knowledge protocol in the standard model is that it can be used to register public keys with a certification authority and bootstrap the system. Once the PKI is available, then one can use the very efficient concurrent zero knowledge proofs in the PKI model.

1.1 This work

The protocol and analysis of [31, 26, 30] are general plausibility results, showing that any language in NP has a concurrent zero knowledge protocol with a logarithmic number of rounds. Although the number of rounds is relatively small (in fact, within a $\omega(\log\log n)$ factor from the optimal for black-box simulation), the steps of the protocol use general results about zero knowledge proof systems for any NP problem. Thus, the resulting protocol is not practical. The goal of this paper is to show that for a large class of languages, the ideas in [31, 26, 30] lead to concurrent zero knowledge proofs that are efficient enough to be practical, i.e., their cost is comparable (within any $\omega(\log n)$ factor) with that of number theoretic operations commonly used in public key cryptography. We show that any language that admits a public coin honest verifier zero-knowledge proof system, can be efficiently transformed into a very strong proof system. The new proof system is *concurrent* zero knowledge with respect to *any* (possibly cheating) verifier via black box simulations. The cost of the transformation is minimal: $\omega(\log n)$ additional rounds (which, by [5], is close to optimal for black-box simulation), where the cost of each additional round is essentially the same of a standard public key operation (say, a modular exponentiation in a finite field). The computational overhead for the remaining rounds is also $\omega(\log n)$ exponentiations for each pair of rounds of the original protocol. Moreover, the soundness and completeness error of the new protocol are essentially the same as the original one. Our protocols are based on a perfectly binding commitment scheme based on the Decisional Diffie-Hellman (DDH) assumption which satisfies some special properties. Note that our transformation works for many interesting protocols. In fact, many of the known zero-knowledge proof systems are public-coin (see for example [22, 20]). Note also that parallel repetition may be used with these protocols to reduce error since we only require *honest verifier* zero knowledge.

A weaker result that follows from our technique is a transformation of computational public-coins *honest-verifier* zero-knowledge proofs into computational public-coins zero-knowledge proofs that are good also for non honest verifiers (in the non-concurrent setting). Such a transformation trivially follows from the fact that everything provable is provable in computational zero-knowledge [20, 23, 2]: one can simply disregard the given public-coins honest-verifier zero-knowledge proof system, and construct a new computational (general) zero-knowledge proof system for the same language from scratch. However, this general (trivial) transformation is not efficient because it requires performing a reduction to a complete problem. Methods for improving the efficiency of the transformation to remove the honest-verifier restriction for computational zero-knowledge protocols have been investigated in [24] and can be obtained from the techniques in [6], but none of these results makes a practical protocol with a widely acceptable security assumption. Our techniques allow such a transformation for public coin zero-knowledge proofs with low overhead and building on the Decisional Diffie Hellman assumption. Note that a similar transformation from honest verifier to cheating verifier for *statistical* zero knowledge does not follow from general completeness results, yet, [21] shows that such transformation is possible in principle. Our transformation is much more efficient than the one in [21], but it does not preserve statistical zero knowledge, i.e., even if applied to a honest verifier statistical zero knowledge proof system, the transformed protocol will satisfy only computational zero knowledge.

Our protocols are based on a new primitive that we denote *simulatable commitment schemes.* These are commitment schemes that satisfy the standard requirements of commitment schemes with respect to secrecy and binding of a commitment phase and a reveal phase. We require two extra features of simulatable commitments. First, we require the existence of a proof system to show, given a pair of strings (c, v), that c is a commitment to the value v. Second, we require an efficient simulator for the view of the honest verifier in this proof system. The definition and construction of an efficient simulatable commitment scheme based on the DDH assumption are provided in Section 4.

It is worth noting the exciting result of Barak [1] presenting a constant round zero-knowledge proof with bounded concurrency. His zero-knowledge proof is a breakthrough in showing that a non trivial non black-box zero-knowledge proof exists. However, this protocol works in the model of bounded concurrency. A polynomial bound on the number of concurrent sessions must be known *a priori* and the length of each message is larger than this number. Thus, for defending against a large polynomial number of concurrent sessions (say, n^3), we need a seemingly inefficient protocol with very long messages. From a practical standpoint we could not find a way to instantiate Barak's protocol in an efficient manner. We believe it is an interesting open question to find a way to make the constant-round zero-knowledge proof of Barak efficient enough to be used in practice.

1.2 Alternative protocols

We are aware of two related protocols that may be used for obtaining concurrent zero-knowledge proof systems without requiring the usage of general NP reductions. The first is a protocol for resettable zero-knowledge that appeared in (the earliest version of) [4], and the other is the protocol of [30]. Both protocols are stated for a specific NP language, but may be used with other similar zero-knowledge public coins protocols. The class of zero knowledge protocols adequate for use with these techniques includes many of the known public coins zero-knowledge protocols (e.g., protocols where the simulator can be used to produce distributions indistinguishable from the *conditional* view of the verifier *given its random coins*), but it does not include the class of all honest verifier (public coins) zero-knowledge protocols. In order to explain the problems that arise when applying these transformations to generic public-coins honest-verifier protocols, we consider a different, but related, transformation that appears to be part of the cryptography community folklore. The folk theorem says: in order to transform a public-coins honest-verifier zero-knowledge protocol into one secure against possibly cheating verifiers (in the non-concurrent setting), let the verifier commit to its random queries at the beginning of the protocol, and then, in place of sending a random queries, open the initial commitments. This transformation (used for example in [19] to preserve zero-knowledge under parallel composition, following a suggestion from [20]) does not by itself enforce honest verifier behavior. Consider for example a honest-verifier proof system where, if the verifier sends the all 0's query the prover reveals some secret. This proof system can still be honest-verifier zero-knowledge because the chances that a honest verifier will send random query 0^n are negligible. Still, letting the verifier commit to the query beforehand does not keep a cheating verifier from making the query 0^n, and extract some secret information from the prover.

It is possible to use some additional tricks (e.g., a two party coin tossing protocol) to make the constructions implicit in [4, 30] work also with any HVZK protocols, thus providing an alternative to the protocol in this paper. However, these two constructions are much less efficient. The first one seems to require (even if adapted to state of the art work on concurrent zero-knowledge) a large number of commitments and it is based on the **strong** DLP assumption. The later protocol is also inefficient requiring $O(\ell^2)$ commitments (for ℓ being the number of rounds in the preamble). The construction in this paper requires only $4(k+1)\ell$ additional exponentiations, where k is the number of rounds in the honest verifier zero-knowledge protocol used (typically, $k = 3$).

1.3 Organization

The rest of the paper is organized as follows. In section 2 we introduce standard definitions and notation. In section 3 we describe the original concurrent zero-knowledge proof that we build on. In section 4 we define the special properties of simulatable commitments to be used in the main protocol, and describe a specific construction of simulatable commitments based on the Decisional Diffie Hellman assumption. In Section 5 we present the main result, showing how (efficient) simulatable commitments can be used to efficiently transform any public coin honest verifier zero knowledge protocol into a concurrent general zero knowledge one. In Section 6 we present the easier, yet interesting, transformation from any public coins honest verifier zero knowledge proof into a zero knowledge proof that is robust also against non honest verifiers.

2 Preliminaries

Due to lack of space, we refrain from presenting standard definitions such as zero-knowledge, auxiliary input zero-knowledge, black box simulation, and witness indistinguishability. We now present some issues that are more within the focus of this paper. In addition to the very basic definitions, we would like to stress the variants that are specifically interesting for us.

Public coin proofs: We say that a proof system is public coins if all the messages of the (honest) verifier are computed by choosing uniformly at random an element in a predetermined set and sending the chosen element to the prover. The power of verification stems (only) from the fact that future verifier messages (challenges) are not known to the prover before the verifier sends them.

The number of rounds: An interaction proceeds in *rounds* of communication. One of the parties sends a message to the second party, and then the other party responds. This goes on until the protocol ends. Each message sent is one round of the protocol. In particular, we will discuss 3 round proofs in which the prover sends one message, the verifier responds, and then the prover finishes with a last message.

Concurrent zero-knowledge. Following [13], we consider a setting in which a polynomial time adversary controls many verifiers simultaneously. However, since in this paper no timing assumption is needed, we slightly simplify the model of [13] and omit any reference to time. Without loss of generality we also assume that messages from the verifiers are immediately answered by the prover. In other words, we assume that the conversation between the prover P and the verifiers $V_1, V_2, \ldots$ is of the form $v_1, p_1, v_2, p_2, \ldots, v_n, p_n$ where each v_j is a message sent by some verifier V_{i_j} to P, and the following p_j is a message sent by P to V_{i_j} in reply to v_j. The adversary $\mathcal{A}$ takes as input a partial conversation transcript, i.e., the sequence of messages received from the prover $p_1, \ldots, p_k$ so far (with the verifiers' messages v_j and their senders V_{i_j} being implicitly specified by the adversarial strategy $\mathcal{A}$). The output of $\mathcal{A}$ will be a pair (i, v), indicating that P receives message v from a verifier V_i. The view of the adversary on input x in such an interaction (including all messages, and the verifiers random tapes) is denoted $(P, \mathcal{A})(x)$.

Definition 1. *We say that a proof or argument system (P, V) for a language L is (computational) black box* concurrent zero-knowledge *if there exists a probabilistic polynomial time oracle machine S (the simulator) such that for any probabilistic polynomial time adversary $\mathcal{A}$, the distributions $(P, \mathcal{A})(x)$ and $S^{\mathcal{A}}(x)$ are computationally indistinguishable for every string x in L.*

In what follows, we will usually refer to the adversary $\mathcal{A}$ as the *adversarial verifier* or the *cheating verifier*, and denote it by V^*. All these terms have the same meaning.

Commitment schemes. We include a short and informal presentation of commitment schemes. For more details and motivation, see [18]. A commitment scheme involves two parties: The *sender* and the *receiver*. These two parties are involved in a protocol which contains two phases. In the first phase the sender commits to a bit (or, more generally, an element from some prescribed set), and in the second phase it reveals it. We make two security requirements which (loosely speaking) are:

Secrecy: At the end of the *commit phase*, the receiver has no knowledge about the value committed upon.

Binding property: It is infeasible for the sender to pass the commit phase successfully and still have two different values which it may reveal successfully in the reveal phase.

Various implementations of commitment schemes are known, each has its advantages in terms of security (i.e., binding for the prover and secrecy for the receiver), the assumed power of the two parties etc. Two-round commitment schemes with perfect secrecy can be constructed from any claw-free collection (see [18]). It is shown in [3] how to commit to bits with statistical security, based on the intractability of certain number-theoretic problems. Dåmgard, Pedersen and Pfitzmann [10] give a protocol for efficiently committing to and revealing strings of bits with statistical security, relying only on the existence of collision-intractable hash functions. Commitment schemes with perfect binding can be constructed from any one-way functions [28].

We will employ different commitment schemes for the prover and the verifier. The prover's scheme will be perfectly binding. In particular, in this work we construct commitment schemes that are perfectly binding and computationally secure with extra special properties. The details are given in Section 4. For perfectly hiding commitments used by the verifier, no special property is needed, and any scheme can be used. (We suggest using Pedersen's commitment, as this scheme can be implemented using the same global parameters as required by our simulatable commitment.)

The commitment scheme that we will use for the prover is non interactive, meaning that the commit phase consists of a single message from the prover to the verifier. The commitment message used to commit to value v using randomness r is denoted $\textsc{commit}_r(v)$. The canonical decommitment procedure is also non interactive, and consist in revealing the randomness r used to compute the commitment message c to the verifier, who checks that c is indeed equal to $\textsc{commit}_r(v)$.

3 The Richardson-Kilian protocol

Richardson and Kilian [31], following ideas of Feige, Lapidot, and Shamir [16], have proposed a concurrent zero-knowledge proof system, for any language in NP, with a polynomial number of rounds. Kilian and Petrank [26] have drastically improved the analysis of the protocol by presenting a new simulation technique and showing that a polylogarithmic number of rounds suffices. The analysis of the Kilian-Petrank simulator has been further improved by Prabhakaran, Sahai, and Rosen [30] showing that the number of rounds can be reduced to any superlogarithmic function $\omega(\log n)$.

The protocol itself has the following structure. Initially the verifier V commits to random values $v_1, \ldots, v_\ell$. Then P and V alternate ℓ times, with P first committing to some value v'_i, and then V revealing v_i opening the corresponding commitment sent in the first round. The intuition is that P tries to guess the value of v_i before V decommits. However, since the commitment protocol used by V is statistically hiding, the prover has only a negligible chance at making the right guess $v'_i = v_i$ for any $i = 1, \ldots, \ell$. After P has committed to $v'_1, \ldots, v'_l$ and the verifier has successfully decommitted $v_1, \ldots, v_l$, P proves in zero knowledge that either $v'_i = v_i$ for some i *or* $x \in L$. More precisely, if $c_1, \ldots, c_\ell$ are the commitments to $v'_1, \ldots, v'_\ell$, P and V engage in a zero knowledge (in fact, witness indistinguishable is enough) proof that the string $x' = (c_1, \ldots, c_\ell, v_1, \ldots, v_\ell, x)$ belongs to the NP language L' of all strings such that either $x \in L$ or $\textsc{commit}(v_i, r_i) = c_i$ for some i and r_i. The intuition for this second stage is that in a real interaction between the prover and the verifier, the chances of P correctly guessing a commitment c_i to the right string v_i before v_i is revealed is negligible. So, proving $x' \in L'$ is essentially equivalent to showing $x \in L$. However, a simulator with black box access to the verifier strategy V^* can produce a conversation between P and V^* by first choosing random values for v'_i, and after some v_i is revealed, "rewind" V^* back to a point after the initial commitment of the verifier to $v_1, \ldots, v_\ell$, but before the commitment of P to c_i (e.g., right after the verifier reveals v_{i-1}). During this second run, when the adversarial verifier reveals v_{i-1}, the simulator replies with a commitment to v_i (as revealed by V^* in the previous run). So, by the time the first stage of the protocol is over, the simulator knows a witness of $x' \in L'$ (namely, the randomness used to commit to $v'_i = v_i$), and can successfully prove the statement $x' \in L'$ to the verifier. This is different from the witness used in a real interaction between P and V, but, because of the witness indistinguishability property of the proof system used in the second stage, the conversation transcript produced by the simulator will be indistinguishable from the real one. As a matter of terminology, the first and second stage of the proof are usually called the *preamble* and the *body* of the proof. The difficulty in carrying this simulation scheme is that the adversarial verifier can cause the simulator to fail by aborting the execution of the verifier protocol before revealing v_i during the first run (in which case the prover is also entitled to abort the protocol), but successfully decommitting v_i during the second run, causing the simulator to reach the body of the proof without knowing an NP-witness for $x' \in L'$. In [5] it is shown that if the number of rounds (in any concurrent black-box zero-knowledge proof system) is $o(\log k / \log\log k)$, then by coordinately aborting several concurrent executions of the protocol, the adversarial verifier can force the simulator to perform a superpolynomial number of rewinding operations in order to simulate the conversation for a non trivial language L. (Namely they show that any polynomial time simulator can be transformed into a probabilistic polynomial time decision procedure for L, showing that L is in BPP.)

In a sequence of papers [31, 26, 30], Richardson, Kilian, Petrank, Prabhakaran, Sahai and Rosen show that if the number of rounds in the preamble is set to any superlogarithmic function $\omega(\log k)$ in the security parameter, then there is a polynomial time rewinding strategy that always allows the simulator to reach the second stage of any concurrent execution of the protocol with a valid commitment to $v'_i = v_i$ for some i. Moreover, the rewinding strategy is independent (oblivious) of the adversarial verifier strategy. It follows (using standard hybrid techniques, and the secrecy properties of commitments and witness indistinguishable proofs) that the final transcript used by the simulator is indistinguishable from a real conversation.

3.1 How we improve the proof system

Our protocol is based on the protocol of [31, 26, 30]. In particular, we use a similar structure of proof system with an important modification. Our proof of correctness relies on the proof in [26] with an additional analysis. In particular, our construction is based on a special commitment scheme such that committed values can be efficiently proved in zero knowledge, i.e., the proof of the commitment can be simulated. Interestingly, our proof system uses the simulator of the commitment scheme as a subroutine, while the concurrent simulator uses the actual proof system (of the commitment scheme). This provides an interesting application of zero-knowledge simulation where the simulator (of the commitment scheme) is not only used in the proof of security of the application, but it is actually invoked by the protocol. Thus, the efficiency of the application directly depends on the efficiency of the simulator.[3]

The main differences between our proof system and the proof systems in [31, 26, 30] is that we use a specific commitment scheme with some interesting properties and that we do not invoke a general zero-knowledge proof for an NP-Complete problem. Instead, we start with any public coins zero-knowledge proof and extend it in an efficient manner making it concurrent. Furthermore, it is enough for us that the original proof system is **honest-verifier** zero-knowledge whereas the Kilian-Petrank transformation required a witness indistinguishable proof system that is good for any verifier.

It should be noted that the number of rounds in the protocols obtained applying our transformation depends on the number of rounds in the original public coin protocol. So, our protocol can have a larger number of rounds than protocols obtained invoking general results for NP (which employ a constant round protocol in the proof body, e.g., 5 rounds in the case of [19]). However, public coin HVZK protocols usually have only a small constant number of rounds (typically 3). So, for most cases of practical interest the round complexity of our protocols is comparable with (or even slightly better than) that of general results for NP. (We remark that since we only need protocols that are zero-knowledge with respect to the honest verifier, the soundness error can be made arbitrarily small by parallel repetition, without increasing the number of rounds of the original protocol.) More importantly, since our transformation does not invoke Cook's theorem, our protocols are much more efficient than protocols obtained from general results from a computational point of view. Details follow. Consider an NP language L. General completeness results immediately give a computational zero knowledge proof system that operates as follow. Let f be a polynomial time computable reduction from L to an NP-Complete problem C for which a zero-knowledge proof system is known. (E.g., the zero knowledge proof system for 3-colorable graphs of [19] used by [26, 30].) In order to prove that $x \in L$, both the prover and the verifier apply function f to x to obtain an instance $f(x)$ of problem C. Finally the known proof system for NP-Complete language C is used to prove that $f(x) \in C$. Even if the proof system for C is reasonably efficient, this scheme hides a big computational overhead behind the application of the reduction function f. Typical reduction functions f perform some sort of gate-by-gate transformation, starting from the verification circuit for the instance-witness relation associated to NP language L. So, the size of the output of f is usually much bigger than the size of the original problem. In contrast, in our proof system the prover algorithm of the original public coin protocol is run unmodified, and the only overhead is a small number of additional public key operations. So, if we start from an efficient public coin protocol, the transformed protocol is also efficient enough to be run in practice.

[3] Such use of simulators within cryptographic protocols is not new, and it has occurred before for example in [8, 11].

4 Simulatable Commitments

We start by defining and constructing simulatable commitment schemes that satisfy some special properties. We will later use these commitment schemes for the efficient transformation from public coin honest verifier proofs into concurrent general zero knowledge proofs.

Simulatable commitment schemes satisfy the standard requirements of (non-interactive) commitment schemes with respect to secrecy and binding of a commitment phase and a reveal phase. In this section, we will be interested in commitments with perfect binding and computational secrecy properties.

We require two extra features of simulatable commitments. First, we require the existence of a proof system to show, given a pair of strings (c, v), that c is a commitment to the value v. Second, we require a simulator for this proof system with some special properties. Let us start with the proof system. The prover (in this proof system) gets as an auxiliary input the randomness used to compute the commitment message, i.e., the string r such that $c = \textsc{commit}_r(v)$. Informally this proof system has the following five properties:

- *3 Rounds:* The prover sends the first message and the last. The verifier sends the second message. We denote these three messages by (m, q, a). (Intuitively: message, query, answer.)
- *Public Coins:* The proof system is "public coin", meaning that the honest verifier chooses its message q uniformly at random from some prescribed set $\mathcal{Q}$.
- *Perfect Completeness:* If the input (c, v) satisfies the property that c is a commitment on v, then the prover produces a first message m such that for any possible verifier choice of $q \in \mathcal{Q}$ the prover continues by outputting an a such that the verifier accepts the proof (m, q, a).
- *Optimal Soundness:* We say that the soundness error of the proof is ϵ if for any common input (c, v) that does not satisfy the property that c is a commitment on v the following holds. For any possible m, there are at most $\epsilon \cdot |\mathcal{Q}|$ strings q that can be answered by the prover. Namely, for at most $\epsilon \cdot |\mathcal{Q}|$ strings q there exists a string a such that the verifier accepts (m, q, a). We say that the soundness is *optimal* if for any possible m there is only one single $q \in \mathcal{Q}$ that can be answered by the prover.
- *Efficiency:* The prover can be implemented as a polynomial time machine given a proof r that c is a commitment on v. Namely, r is a string such that $c = \textsc{commit}_r(v)$.

Our second requirement of a simulatable commitment scheme is that there exists a (non-rewinding) simulator S for the view of the honest verifier in the above proof. We call a pair of strings (c, v) *legitimate input* if v is any string and c is a possible commitment $c = \textsc{commit}_r(v)$ to value v for some r. The following two distributions are polynomially indistinguishable over the set of legitimate pairs (c, v):

1. *Interactions with the honest verifier:* Invoke the prover on input (c, v, r) (and uniformly chosen random coin tosses) to produce a first proof message m,

choose uniformly at random a query $q \in \mathcal{Q}$ and invoke the prover again with the chosen query q to get the third message a. The output is set to the computed triplet (m, q, a).
2. *Simulation of legitimate inputs:* the output of simulator S on input (c, v) (and uniformly chosen random coin tosses).

Having stated the properties of the simulator, let us make an important claim about the output distribution of the simulator on "bad" inputs.

Claim. The following two distributions are polynomially indistinguishable over strings v in the domain of committable values.

- *Simulation of random commitments $\Omega_1(v)$:* Invoke the commitment scheme on v to get $c = \textsc{commit}_r(v)$ (using uniformly chosen random coin tosses r), invoke the simulator on input (c, v) (and uniformly chosen random coin tosses), and output the resulting triplet (m, q, a).
- *Simulation of random bad commitments $\Omega_2(v)$:* Choose uniformly at random a value v' from the domain of committable values, invoke the commitment scheme on v' to get $c = \textsc{commit}(v')$ (using uniformly chosen random coin tosses), invoke the simulator on input (c, v) (and uniformly chosen random coin tosses), and output the resulting triplet (m, q, a).

Sketch of proof: If the distributions $\Omega_1(v)$ and $\Omega_2(v)$ are polynomial time distinguishable then it contradicts the secrecy property of the commitment scheme. In order to check with polynomial advantage whether a commitment c is a valid commitment on a value v one may run the simulator on input (c, v) and use the given distinguisher to get a polynomial advantage. This contradicts the secrecy property of the commitment scheme. □

To allow future reference to the prover machine, verifier machine and simulator, we adopt the following notations. The proof system is specified by a tuple $(P^0_{com}, P^1_{com}, V_{com}, \mathcal{Q}, S_{com})$ where

- (P^0_{com}, P^1_{com}) is the prover strategy. More precisely, P^0_{com} is a probabilistic polynomial time algorithm that on input (c, v, r) such that $\textsc{commit}_r(v) = c$, outputs the first prover message m and some state information s. On input (c, v, r), state information s and challenge q, P^1_{com} outputs an "answer" a to challenge q.
- $\mathcal{Q}$ is the set of possible challenges and V_{com} is a (deterministic) polynomial time verification procedure that on input a transcript (c, v, m, q, a) either accepts or rejects the input.
- S_{com} is a probabilistic polynomial time algorithm that on input a pair (c, v) of a value v and a commitment c, outputs a transcript (m, q, a) distributed as described above.

4.1 Using simulatable commitments

We will build a commitment scheme with optimal soundness. Let us point out an important feature of this scheme, that allows this proof system to serve as a building block in our zero-knowledge proof. If the common input is (c, v) and c is **not** a commitment to v, then once m is fixed, a matching q is completely determined. No other $q \in \mathcal{Q}$ can be completed into a convincing proof (m, q, a) because of the optimal soundness property. On the other hand, if c **is** a commitment to v, then fixing m has no influence over q. Any $q \in \mathcal{Q}$ can be completed into a triple (m, q, a) that convinces the verifier.

A polynomial time algorithm cannot tell whether a pair (c, v) is legitimate or not. Thus, the verifier or a distinguisher that looks at a proof transcript cannot tell if m determines q (i.e., the pair was bad) or the choice of m allows any $q \in Q$ to be used next (i.e., the pair was good). The fact that any $q \in Q$ can be chosen (for good input pairs) will give the prover an advantage: the power to select any value $q \in \mathcal{Q}$ that can be answered, after m is fixed. Such *influence games* in which the prover has more influence if the input is legitimate, and less influence if it is not, have been used in previous zero-knowledge protocols. See for example [8, 11, 25].

We will now show how to obtain simulatable commitment schemes, and then proceed with using simulatable commitment to implement efficient concurrent zero knowledge proof systems (see Section 5 below).

4.2 Commitment schemes based on DDH

In this section we construct a simulatable commitment scheme and prove its properties under the DDH assumption. We show that our commitment scheme admits a very efficient zero knowledge proof system with perfect completeness and optimal soundness, with a very efficient simulator. As noted earlier, in our application we will use the simulator of the proof system associated to the commitment scheme to build a concurrent zero knowledge protocol. Therefore, in order to get an efficient protocol, it is not enough to have a simulator that runs in polynomial time, but we need a simulator that is also reasonably efficient in practice.

Our scheme is based on exponentiation in finite groups, but it is quite different from other discrete logarithm based commitment schemes, like Pedersen's [29].[4] We assume a finite group G of large prime order Q such that the DDH problem in G is hard. We also assume that random elements of G can be efficiently generated, and membership in G can be decided efficiently.[5] We remark

[4] In fact, Pedersen's algorithm is statistically hiding, and only computationally binding, so it would allow a computationally unbounded prover to cheat. Moreover, Pedersen's commitment does not have the simulatability property required by our application. Another commitment scheme based on discrete exponentiation is the *trapdoor* commitment of [17]. As Pedersen's, this commitment scheme is only computationally binding. Moreover, the scheme only allows to commit to single bit messages. On the other hand, like our scheme, the trapdoor commitment of [17] has some special property (the trapdoor) that makes it useful in their application.

[5] Groups with the properties above are standard in cryptography. For concreteness, we exemplify a specific construction. The group G can be specified by two sufficiently large primes P, Q such that Q divides $P - 1$. The group G is the order Q subgroup

that the hardness of the DDH problem in G is only used to protect the sender from a cheating receiver. So, one can let the sender choose group G and send its description to the receiver. In the sequel, we consider the group G (and its order Q) as fixed, and all arithmetic operations and equalities should be interpreted in group G, or modulo Q if they occur at an exponent.

In our commitment scheme, the prover first chooses two random elements g, h of $G \setminus \{1\}$. Elements g, h are sent to the receiver together with the actual commitment, and the receiver is responsible for checking that g and h generate G.[6] However, the same g, h can be reused for many commitments. Parameters G, Q, g and h can be used to commit to any value in $\mathcal{Q} = \{0, \ldots, Q-1\}$. The commitment function is $\text{COMMIT}_r(v) = (g^r, h^{r+v})$, where $v, r \in \mathcal{Q}$. Upon receiving the commitment $(\bar{g}, \bar{h})$, the verifier checks that $\bar{g}, \bar{h}$ belong to G. If this is the case, then $(\bar{g}, \bar{h})$ can certainly be expressed as (g^r, h^{v+r}), for some (possibly unknown to the verifier) r and v, because g and h are generators for G.[7]

Perfect binding: The binding property immediately follows (in an information theoretic manner) from the fact that g and h have order Q. Therefore, g^r, h^{r+v} uniquely determine the values of r and $r + v \pmod{Q}$, and so also the value of v.

Computational secrecy: The computational secrecy property immediately follows from the DDH assumption. Informally, since g is a generator it is possible to write $h = g^\omega$ for some $\omega \in \{0, \ldots, Q-1\}$. Using this fact, the DDH assumption implies that it is impossible to distinguish (efficiently) between (g, h, g^r, h^r) for a uniform $r \in \{0, \ldots, Q-1\}$ and (g, h, g^{r_1}, h^{r_2}) for uniform r_1 and r_2 in $\{0, \ldots, Q-1\}$. Computational secrecy follows. Formally, one has to use the self reducibility of the DDH problem. The details are omitted.

The proof system. We present a proof system to prove that $(\hat{g}, \hat{h})$ is a commitment to v, i.e., that $(\hat{g}, \hat{h}) = (g^r, h^{r+v})$ for some r. We specify a 3-round public coin proof system using the procedures $P^0_{com}, P^1_{com}, \mathcal{Q}, V_{com}$ introduced in the previous section. The common input to the proof is a tuple consisting of global parameters G, Q, g, h, commitment string $(\hat{g}, \hat{h})$ and value v. The prover also gets NP-witness r as an auxiliary input. The query set is the same as the set of input values $\mathcal{Q}$. In the first stage the prover $P^0_{com}((\hat{g}, \hat{h}), v, r)$ outputs message $(\bar{g}, \bar{h}) = (g^s, h^s)$ and state information s, where $s \in \mathcal{Q}$ is chosen uniformly at random. The answer $P^1_{com}((\hat{g}, \hat{h}), v, r, s, q)$ of the prover to query $q \in \mathcal{Q}$ is $a = qr + s \pmod{Q}$. The verification procedure $V((\hat{g}, \hat{h}), v, (\bar{g}, \bar{h}), q, a)$ accepts if and only if $\hat{g}^q \bar{g} = g^a$ and $(\hat{h}/h^v)^q \bar{h} = h^a$.

of Z^*_P. These parameters can be chosen by the prover at random, and the verifier simply checks that P, Q are prime, and Q divides $P-1$. Membership of g in G can be easily decided checking that $g^Q = 1 \pmod{P}$. Finally, group G can be efficiently sampled by picking a random g' in Z_P, and computing $g = (g')^{(P-1)/Q} \pmod{P}$.

[6] This is easily done checking membership $g, h \in G$ and $g, h \neq 1$. Since group G has prime order, it follows that g and h are both generators for G.

[7] This check is not essential for the standard commitment and decommitment operation, but it will be useful to give a proof system for the simulatable commitment.

It is easy to verify that the proof system has perfect completeness, i.e., if the prover follows the protocol it can answer any query $q \in Q$ so as to make the verifier accept, and all procedures run in polynomial time. We now argue that the proof has optimal soundness. Assume that for some $\bar{g}, \bar{h}$, there are two (distinct) q_1, q_2 for which the prover can make the verifier accept, with answers a_1, a_2 respectively. Since $\hat{g}$ and $\hat{h}$ belong to G we can write $(\hat{g}, \hat{h}) = (g^r, h^{r+v'})$ for some r, v'. We want to prove that $v' = v$. From the last check performed by the verifier we know that $\hat{g}^{q_i}\bar{g} = g^{a_i}$ and $(\hat{h}/h^v)^{q_i}\bar{h} = h^{a_i}$ for $i = 1, 2$. It follows that $\hat{g}^{q_1-q_2} = g^{a_1-a_2}$ and $(\hat{h}/h^v)^{q_1-q_2} = h^{a_1-a_2}$. Using $h = g^\omega$ and $(\hat{g}, \hat{h}) = (g^r, h^{r+v'})$, we get

$$\begin{aligned} g^{r\omega(q_1-q_2)} &= \hat{g}^{(q_1-q_2)\omega} = g^{(a_1-a_2)\omega} = h^{(a_1-a_2)} \\ &= (\hat{h}/h^v)^{(q_1-q_2)} = g^{\omega(r+v'-v)(q_1-q_2)}. \end{aligned}$$

Taking the quotient of the first and last term, and extracting the discrete logarithm to the base g, we get $v(q_1 - q_2) = v'(q_1 - q_2) \pmod{Q}$. So, if $q_1 \neq q_2$, then $(q_1 - q_2)$ is invertible modulo Q (because $0 \leq q_1, q_2 < Q$), and dividing by $(q_1 - q_2)$ yields equality $v = v'$.

The simulator. It remains to show that there exists a simulator $S_{com}((\hat{g}, \hat{h}), v)$ with the required properties (see Section 4). Here we describe the simple simulator. On input $(\hat{g}, \hat{h})$ and v, the simulator chooses $q, a \in \{0, \ldots, Q-1\}$ independently and uniformly at random and sets $\bar{g} = g^a/\hat{g}^q$, $\bar{h} = h^a/(\hat{h}/h^v)^q$. The output of the simulator is $((\bar{g}, \bar{h}), q, a)$. To see that the simulator outputs the correct distribution, we first check how the distribution of interactions of the prover with the honest verifier looks like. This distribution consists of triplets of the form $((g^s, h^s), q, a)$, where s and q are chosen uniformly at random in $\{0, \ldots, Q-1\}$ and a is set to $a = q \cdot r + s \pmod{Q}$. The same distribution is obtained if one chooses $q, a \in \{0, \ldots, Q-1\}$ independently and uniformly at random, and then sets $s = a - q \cdot r \pmod{Q}$ and outputs $((g^s, h^s), q, a)$. The latter is the output distribution of the simulator on legitimate inputs. Thus, the simulator perfectly simulates the view of the honest verifier for any legitimate pair (c, v), and the simulator does not rewind the verifier, as required by the definition of simulatable commitments.

5 Efficient Concurrent Zero Knowledge

In this section we show that any public coin honest verifier zero knowledge proof system can be efficiently transformed into a new proof system which is *concurrent* zero knowledge with respect to *any* verifier. The transformation is based on any simulatable and perfectly binding commitment scheme.

5.1 An overview

We modify the concurrent zero-knowledge proof system of [31, 26, 30] in the following manner. We start with a similar preamble. The verifier begins by committing to ℓ random strings $v_1, \ldots, v_\ell$, where ℓ is any function asymptotically larger than $\log n$. The verifier may use any efficient statistically hiding commitment scheme. As in the protocols of [31, 26, 30], the parties repeat for $i = 1, \ldots, \ell$ the following steps: the prover commits to a random string v'_i and then the verifier reveals the string v_i opening the corresponding commitment sent in the first round. The prover uses a simulatable commitment scheme for his commitments. The real prover cannot hope to set $v'_i = v_i$, except with negligible probability. On the other hand, as shown in [31, 26, 30], the rewinding simulator may set $v'_i = v_i$ for one of the rounds $1 \leq i \leq \ell$.

Next comes the body of the proof in which the prover shows that the input is in the language. Here we provide a mechanism by which the prover and the verifier together generate a random string to be used as the verifier's challenge in the original protocol. This random string is guaranteed to be uniformly chosen at random if both of the following two conditions hold:

1. The prover has not managed to set $v'_i = v_i$ for any round $1 \leq i \leq \ell$ in the preamble, and,
2. One of the parties (either the prover or the verifier) follows the protocol.

On the other hand, if $v'_i = v_i$ for some i, then the prover has the power to set the outcome of the coin tossing protocol to any value of its choice. The random string output by this protocol is used to run the original public coins proof and show that the common input is in the language.

Completeness will easily follow. To show that soundness holds, we note that the prover can only break the first condition with negligible probability, and since the verifier follows the protocol, the random tape used is indeed random. Therefore, soundness follows from the soundness property of the original protocol.

To claim zero-knowledge we note that the simulator may set $v'_i = v_i$ for one of the rounds $1 \leq i \leq \ell$ even in the concurrent setting. Using this, the simulator gets control over the choice of the coin tosses for each of the rounds in the body of the protocol. Therefore, it can use the following strategy. Run the simulator of the original (honest verifier, non concurrent) proof system. This yields a transcript of the body of the proof that is indistinguishable from the actual original interactive proof. The simulator then forces the body of the concurrent proof be this output of the original simulation. Doing this requires control over the choice of the verifier random coin tosses in the simulated protocol, which he has, given that $v'_i = v_i$.

We proceed with a formal description of the protocol and simulation.

5.2 The protocol

Let COMMIT be a simulatable perfectly binding commitment scheme, and let $(P^0_{com}, P^1_{com}, V^1_{com}, \mathcal{Q}, S_{com})$ be the corresponding proof system and simulator. Let also (P_L, V_L) be any public coin honest verifier zero knowledge proof system for some language L. Let $2k + 1$ be the number of rounds of the (original) protocol. Without loss of generality we assume that the verifier messages are chosen from a set $\mathcal{Q}$.[8] More precisely, the prover P_L is an algorithm that on input a string $x \in L$, some randomness r_L, and a sequence of messages $q_1, \ldots, q_i$, (for $1 \leq i \leq k$) outputs the next prover message p_{i+1}. The verifier algorithm answers each prover message p_i with a random and independently chosen q_i, and at the end of the interaction applies a verification procedure $V_L(x, p_1, q_1, \ldots, p_k)$ to determine whether to accept or reject x.

We show how to combine (P_L, V_L) and the commitment scheme to design a new interactive proof system for language L. Let $\ell(n)$ be any function such that $\ell(n) = \omega(\log n)$. In what follows $\ell = \ell(|x|)$. The new proof system is the following.

1. The verifier chooses uniformly at random a sequence of values $v_1, \ldots, v_\ell$ from an exponentially large set V. The verifier commits to the values $v_1, \ldots, v_\ell$ in the sequence using a perfectly hiding commitment scheme. The commitments are sent to the prover.
2. The following two steps are executed for $i = 1, \ldots, \ell$
   (a) The prover chooses uniformly at random a value $v'_i \in V$, computes a commitment $c_i = \textsc{commit}_r(v'_i)$ using a simulatable commitment scheme (and fresh randomness r each time), and sends the commitment to the verifier.
   (b) The verifier opens v_i. If at any point the verifier does not open the commitment properly, the prover aborts the execution of the protocol.
3. The following four steps are executed for $i = 1, \ldots, k$. Steps (c) and (d) below can be merged with steps (a) and (b) of the following iteration.
   (a) For all $j = 1, \ldots, \ell$, the prover runs the simulator S_{com} on input (c_j, v_j) to obtain transcript $(m_{i,j}, q_{i,j}, a_{i,j})$. (Informally, here the prover uses the simulator to "pretend" that his guess c_j for v_j was correct.) Then it computes $p_i = P_L(x, r, q_1, \ldots, q_{i-1})$ and sends $(m_{i,1}, \ldots, m_{i,\ell}, p_i)$ to the verifier.
   (b) The verifier picks a randomly chosen q'_i and sends it to the prover.
   (c) The prover computes $q_i = \bigoplus_j q_{i,j} \oplus q'_i$ and sends $q_{i,1}, q_{i,2}, \ldots, q_{i,\ell}$ and $a_{i,1}, a_{i,2}, \ldots, a_{i,\ell}$ to the verifier.
   (d) The verifier checks that $V_{com}(c_j, v_j, m_{i,j}, q_{i,j}, a_{i,j}) = \mathit{accept}$ for all $j = 1, \ldots, \ell$ and computes $q_i = \bigoplus_j q_{i,j} \oplus q'_i$.
4. The prover computes $p_{k+1} = P_L(x, r, q_1, \ldots, q_k)$ and sends p_{k+1} to the verifier.
5. The verifier accepts if and only if V_L accepts $(p_1, q_1, \ldots, q_k, p_{k+1})$.

In what follows, we denote Steps (1) and (2) as the *preamble* of the proof. Step (3) is the *body* of the proof. We now state the main theorem.

Theorem 1. *Let (P_L, V_L) be an interactive proof (or argument) system for a language L such that the proof is honest verifier public coins auxiliary input zero-knowledge proof system for L and such that the prover can be implemented as an*

[8] Shorter messages can be emulated by letting the prover ignore part of the message, while longer messages can be emulated by concatenating several blocks.

efficient machine given a witness to the input being in the language. Then the above protocol is an interactive proof for L with the following properties. (1) If the original protocol (P_L, V_L) *has completeness and soundness errors* (err_c, err_s)*, then the new protocol has errors* $(err_c + \epsilon_1, err_s + \epsilon_2)$ *where* ϵ_1 *and* ϵ_2 *are negligible functions. (2) The prover of the above protocol works in polynomial time given black box access to the prover* P_L *and to the simulator* S_{com}*. In particular, if L is in NP and* P_L *can be implemented efficiently given an NP-witness for* x*, then also the new prover has an efficient implementation given the NP witness. (3) The protocol is concurrent zero-knowledge via black box simulation. The simulator for the new protocol works in polynomial time given black box access to the verifier* V^**, to the simulator* S_L *guaranteed for the view of the honest* V_L *in the original interaction, and to the machines* P_{com}*,* S_{com} *guaranteed for the simulatable commitment scheme.*

For lack of space in this abridged version of the paper, we do not include the proof of the theorem, and the efficiency analysis of the protocol. These two are provided in our full paper.

6 From honest verifier to (non-concurrent) general zero knowledge

A simplification of the above protocol yields a transformation of any honest verifier zero-knowledge proof into a normal zero-knowledge proof, yet without achieving robustness to concurrent composition. Namely, here the goal is to remove the honest verifier restriction for standard zero-knowledge proof systems. The transformation incurs very low overhead on the original protocol. To do this, we note that it is enough to use $\ell = 1$ when concurrent sessions are not considered. Since we can always rewind the verifier without a cost in concurrent sessions, we can always set $v_1' = v_1$ and use it in the body of the proof. The cost of this transformation for an honest-verifier public-coins zero-knowledge interactive proof that has $2k + 1$ rounds is an addition of $4k + 2$ exponentiations. The increase in round complexity is only by 3 rounds. At this low cost, we remove the honest-verifier restriction from any public-coins honest-verifier computational zero-knowledge interactive-proof. The theorem follows.

Theorem 2. *Let* (P_L, V_L) *be an interactive proof system (or argument) for a language L such that the proof is honest verifier public coins auxiliary input zero-knowledge proof system for L and such that the prover can be implemented as an efficient machine given a witness to the input being in the language. Then the protocol of Section 5.2 with* $\ell = 1$ *is an interactive proof for L with the following properties. (1) If the original protocol* (P_L, V_L) *has completeness and soundness errors* (err_c, err_s)*, then the new protocol has errors* $(err_c, err_s + 1/|V|)$*, where* $|V|$ *is superpolynomial in the security parameter. (2) The prover of the above protocol works in polynomial time given black box access to the prover* P_L *and to the simulator* S_{com}*. In particular, if L is in* NP *and* P_L *can be implemented efficiently given an NP-witness for* x*, then also the new prover has an efficient* *implementation given the NP witness. (3) The protocol is zero-knowledge via black box simulation. The simulator for the new protocols works in polynomial time given black box access to the cheating verifier V^*, to the simulator S_L for the view of the honest V_L in the original interaction, and to the prover and simulator protocols P_{com}, S_{com} of the simulatable commitment scheme.*

7 Conclusion

We have shown that any public coin honest verifier zero knowledge protocol can be efficiently transformed into a black box concurrent zero knowledge one. The cost of the transformation is close to optimal: the number of rounds is increased only by an additive term which is an arbitrarily small superlogarithmic function of the security parameter, and the communication complexity of each round is also increased by the same superlogarithmic additive term.

Our solution corresponds to a clever instantiation of the scheme of Kilian, Petrank, and Richardson with a specific commitment scheme and proof system satisfying some special properties, thereby avoiding the use of generic results about zero knowledge proofs for problems in NP, which although polynomial are not practical.

Beside the specific proof system presented in this paper, our construction demonstrates that even generic constructions as the one in [26, 30] that are usually interpreted as mere plausibility results can lead to efficient protocols when properly instantiated.

8 Acknowledgment

We thank the anonymous referees for their deep remarks.

References

1. B. Barak. How to Go Beyond The Black-Box Simulation Barrier. In *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science - FOCS 2001*, pages 106–115, Las Vegas, Nevada, USA, Oct. 2001. IEEE.
2. M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, and P. Rogaway. Everything provable is provable in zero-knowledge. In S. Goldwasser, editor, *Advances in cryptology - CRYPTO '88, Proceedings*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56, Santa Barbara, California, USA, Aug. 1988. Springer-Verlag.
3. G. Brassard, D. Chaum, and C. Crépeau. Minimum Disclosure Proofs of Knowledge. *Journal of Computer and System Sciences*, 37(2):156–189, Oct. 1988.
4. R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. Resettable zero-knowledge. Report TR99-042 (Revision 1), Electronic Colloquium on Computational Complexity (ECCC), Available at URL ftp://ftp.eccc.uni-trier.de/pub/eccc, 2000. Preliminary version in Proc. of STOC 2000.

5. R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-box concurrent zero-knowledge requires $\tilde{\Omega}(\log n)$ rounds. In *Proceedings of the thirty-third Annual ACM Symposium on Theory of Computing - STOC 2001*, pages 570–579, Heraklion, Crete, Greece, July 2001. ACM.
6. D. Chaum, I. Damgård, and J. van de Graaf. Multiparty computations ensuring secrecy of each party's input and correctness of the result. In C. Pomerance, editor, *Advances in cryptology - CRYPTO '87, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 87–119, Santa Barbara, California, USA, Aug. 1987. Springer-Verlag.
7. T. Cohen, J. Kilian, and E. Petrank. Responsive round complexity and concurrent Zero-Knowledge. In C. Boyd, editor, *Advances in Cryptology - ASIACRYPT 2001, Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security*, volume 2248 of *Lecture Notes in Computer Science*, pages 422–441, Gold Coast, Australia, Dec. 2001. Springer-Verlag.
8. I. Damgård and R. Cramer. On monotone function closure of statistical zero-knowledge. Technical Report 1996/003, IACR Cryptology ePrint Archive, Available at URL http://eprint.iacr.org/, 1996.
9. I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In B. Preneel, editor, *Advances in Cryptology - EUROCRYPT 2000, Proceedings of the Internarional Conference on the Theory and Application of Cryptographic Techniques*, volume 1807 of *Lecture Notes in Computer Science*, pages 418–430, Bruges, Belgium, May 2000. Springer-Verlag.
10. I. Damgård, T. P. Pedersen, and B. Pfitzmann. On the existence of statistically hiding bit commitment schemes and fail-stop signatures. *Journal of Cryptology*, 10(3):163–194, 1997. Preliminary version in Proc. of CRYPTO 93.
11. A. De Santis, G. Di Crescenzo, G. Persiano, and M. Yung. On monotone formula closure of SZK. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science 1994 - FOCS '94*, pages 454–465, Santa Fe, New Mexico, USA, Nov. 1994. IEEE.
12. G. Di Crescenzo and R. Ostrovsky. On Concurrent Zero-Knowledge with Pre-processing. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, Proceedings of the 19th Annual International Cryptology Conference*, volume 1666 of *Lecture Notes in Computer Science*, pages 485–502, Santa Barbara, California, USA, Aug. 1999. Springer-Verlag.
13. C. Dwork, M. Naor, and A. Sahai. Concurrent zero-knowledge. In *Proceedings of the thirtieth Annual ACM Symposium on Theory of Computing - STOC '98*, pages 409–418, Dallas, Texas, USA, May 1998. ACM.
14. C. Dwork and A. Sahai. Concurrent zero-knowledge: Reducing the need for timing constraints. In H. Krawczyk, editor, *Advances in Cryptology - CRYPTO '98, Proceedings of the 18th Annual International Cryptology Conference*, volume 1462 of *Lecture Notes in Computer Science*, pages 442–457, Santa Barbara, California, USA, Aug. 1998. Springer-Verlag.
15. U. Feige. *Alternative Models for Zero Knowledge Interactive Proofs*. PhD thesis, Weizmann Institute of Science, Rehovot, Israel, 1990.
16. U. Feige, D. Lapidot, and A. Shamir. Multiple noninteractive zero knowledge proofs under general assumptions. *SIAM Journal on Computing*, 29(1):1–28, 1999. Preliminary version in Proc. of FOCS '90.
17. U. Feige and A. Shamir. Zero knowledge proofs of knowledge in two rounds. In G. Brassard, editor, *Advances in cryptology - CRYPTO '89, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 526–544, Santa Barbara, California, USA, Aug. 1989. Springer-Verlag.

18. O. Goldreich. *Foundation of Cryptography - Basic Tools*. Cambridge Unievrsity Press, 2001.
19. O. Goldreich and A. Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.
20. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(3):691–729, July 1991.
21. O. Goldreich, A. Sahai, and S. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zero-knowledge. In *Proceedings of the thirtieth Annual ACM Symposium on Theory of Computing - STOC '98*, pages 399–408, Dallas, Texas, USA, May 1998. ACM.
22. S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version in Proc. of STOC '85.
23. R. Impagliazzo and M. Yung. Direct minimum-knowledge computations. In C. Pomerance, editor, *Advances in cryptology - CRYPTO '87, Proceedings*, volume 293 of *Lecture Notes in Computer Science*, pages 40–51, Santa Barbara, California, USA, Aug. 1987. Springer-Verlag.
24. J. Kilian. Achieving zero-knowledge robustly. In A. Menezes and S. A. Vanstone, editors, *Advances in Cryptology - CRYPTO '90, Proceedings*, volume 537 of *Lecture Notes in Computer Science*, pages 313–325, Santa Barbara, California, USA, Aug. 1990. Springer-Verlag.
25. J. Kilian and E. Petrank. An efficient noninteractive zero-knowledge proof system for NP with general assumptions. *Journal of Cryptology*, 11(1):1–27, 1998.
26. J. Kilian and E. Petrank. Concurrent and resettable zero-knowledge in polyloalgorithm rounds. In *Proceedings of the thirty-third Annual ACM Symposium on Theory of Computing - STOC 2001*, pages 560–569, Heraklion, Crete, Greece, July 2001. ACM.
27. J. Kilian, E. Petrank, and C. Rackoff. Lower bounds for zero knowledge on the internet. In *Proceedings of the 39th Annual Symposium on Foundations of Computer Science - FOCS '98*, pages 484–492, Palo Alto, California, USA, Nov. 1998. IEEE.
28. M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991. Preliminary version in Proc. of CRYPTO '89.
29. T. P. Pedersen. Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing. In J. Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140, Santa Barbara, California, USA, Aug. 1991. Springer-Verlag.
30. M. Prabhakaran, A. Rosen, and A. Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *Proceedings of the 43rd Annual Symposium on Foundations of Computer Science - FOCS 2002*, pages 366–375, Vancouver, British Columbia, Canada, Nov. 2002. IEEE.
31. R. Richardson and J. Kilian. On the concurrent composition of zero-knowledge proofs. In J. Stern, editor, *Advances in Cryptology - EUROCRYPT '99, Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, volume 1592 of *Lecture Notes in Computer Science*, pages 415–431, Prague, Czech Republic, May 1999. Springer-Verlag.
32. A. Rosen. A note on the round-complexity of concurrent zero-knowledge. In M. Bellare, editor, *Advances in Cryptology - CRYPTO 2000, Proceedings of the 20th annual international Cryptology conference*, volume 1880 of *Lecture Notes in Computer Science*, Santa Barbara, California, USA, Aug. 2000. Springer-Verlag.