Electronic Colloquium on Computational Complexity, Report No. 45 (2002)

Efficient and Concurrent Zero-Knowledge from any public coin HVZK protocol

Daniele Micciancio * Erez Petrank †

July 8, 2002

Abstract

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, independently of the complexity of the original protocol. Moreover, the transformation preserves (up to negligible additive terms) the soundness and completeness error probabilities. The new proof system is proved secure based on the Decisional Diffie-Hellman (DDH) assumption, in the standard model of computation, i.e., no random oracles, shared random strings, or public key infrastructure is assumed. In addition to the introduction of a practical protocol, this construction provides yet another example of ideas in plausibility results that turn into ideas in the construction of practical protocols.

We prove our main result by developing a mechanism for simulatable commitments that may be of independent interest. In particular, it allows a weaker result that is interesting as well. We present an *efficient* transformation of any *honest verifier* public-coin computational zero-knowledge proof into a (public coin) computational zero-knowledge proof secure against *any verifier*. The overhead of this second transformation is minimal: we only increase the number of rounds by 3, and increase the computational cost by 2 public key operations for each round of the original protocol. The cost of the more general transformation leading to concurrent zero knowledge is also close to optimal (for black box simulation), requiring only $\omega(\log n)$ additional rounds (where n is a security parameter and $\omega(\log n)$ can be any superlogarithmic function of n (e.g., $\log n \cdot \log^* n$), and $\omega(\log n)$ additional public key operations for each round of the original protocol.

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 really following the protocol instructions, without revealing any extra information. The original formulation of the notion of zero knowledge [27] considers a single prover and a single verifier working in isolation. This formulation is clearly inadequate in 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 has been the subject of many recent investigations [15, 33, 14, 9, 10, 5, 39, 32, 6, 8, 1]. For example, in [10, 5], 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 [6] have shown that no nontrivial language (i.e., no language

*Computer Science and Engineering Department, University of California, San Diego. La Jolla, California. Email: daniele@cs.ucsd.edu. Research supported in part by NSF Career Award CCR-0093029.

†Dept. of Computer Science, Technion - Israel Institute of Technology, Haifa 32000, Israel. Email: erez@cs.technion.ac.il. This research was supported by the Technion V.P.R. Fund - N. Haar and R. Zinn Research Fund.

ISSN 1433-8092

outside BPP) has constant round black box concurrent zero knowledge proofs. In particular, [6] provides a $o(\log n/\log\log n)$ lower bound on the number of rounds for any such protocol. Interestingly, Kilian, Petrank and Richardson [38, 32] gave an upper bound showing that a polylogarithmic number of rounds is sufficient to achieve concurrent zero knowledge, and very recently Prabhakaran and Sahai [37] improved the analysis of the Kilian-Petrank-Richardson simulator to show that $\omega(\log k)$ many rounds are enough.

Although less efficient than solutions in the PKI model, the solution of [32, 37] 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.

The protocols of [32, 37] are general plausibility results, showing that any language in NP has a concurrent zero knowledge protocol with a (poly)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 [32, 37] 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 [6], 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 [26, 23, 21]).

A weaker result that follows from our technique is a transformation of (non-concurrent) computational public-coins honest-verifier zero-knowledge proofs into computational public-coins zero-knowledge proofs that are good also for non honest verifiers. Such a transformation clearly follows from the fact that everything provable is provable in computational zero-knowledge [23, 28, 2]. However, the general transformation is not efficient. Methods for improving the efficiency of the transformation to remove the honest-verifier restriction for computational zero-knowledge protocols have been investigated in [30] and can be obtained from the techniques in [7], 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 results, yet, [25] shows that such transformation is possible in principle. Our transformation is much more efficient than the one in [25], 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.)

Finally, it is worth noting the exciting result of Barak [1] about concurrent zero-knowledge with constant number of rounds. His zero-knowledge proof is a breakthrough in showing that non black-box zero-knowledge 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 concurrent zero-knowledge proof of Barak efficient enough to be used in practice.

1.1 Organization

The rest of the paper is organized as follows. In section 2 we introduce standard definitions and notation. Some of the standard definitions are relegated to Appendix A. In section 4 we define the special properties of "simulatable commitments" to be used in the main protocol. In Section 5 we describe a specific construction of simulatable commitments based on the Decisional Diffie Hellman assumption. In Section 6 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 7 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. A possible commitment to be used by the verifier is Pedersen's commitment scheme [36], based on the discrete logarithm problem, and described in Appendix B.

2 Preliminaries

We relegate the basic cryptographic preliminaries to the appendix. See Section A.1 for a short reminder of the zero-knowledge concept, Section A.2 for auxiliary input zero-knowledge, Section A.3 for a discussion of black box simulation, and Section A.4 for witness indistinguishability.

2.1 Some notions used in 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.

We go on into the basics of concurrent zero-knowledge and commitment schemes.

2.2 Concurrent zero-knowledge

Following [14], 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 [14] 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 2.1 *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.

2.3 Commitment schemes

We include a short and informal presentation of commitment schemes. For more details and motivation, see [20]. 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. A useful intuition to keep in mind is the "envelope implementation" of bit commitment. In this implementation, the sender writes a bit on a piece of paper, puts it in an envelope and gives the envelope to the receiver. In a second (later) phase, the *reveal* phase, the receiver opens the envelope to discover the bit that was committed on. In the actual digital protocol, we cannot use envelopes, but the goal of the cryptographic machinery used, is to simulate this process.

More formally, a commitment scheme consists of two phases. First comes the *commit* phase and then we have the *reveal* phase. 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 [20]). 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 [11] 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 [34].

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 a special kind of commitment schemes that are perfectly binding and computationally secure with extra special properties. The details are given in Section 4. For perfectly hiding commitments that the verifier will use, we choose a standard technique, see Appendix B. 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 $\text{COMMIT}_r(v)$. 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 $\text{COMMIT}_r(v)$.

3 The KPPRS protocol

Richardson and Kilian [38], following ideas of Feige, Lapidot, and Shamir [17], have proposed a concurrent zero-knowledge proof system, for any language in NP, with a polynomial number of rounds. Kilian and Petrank [32] have drastically improved the analysis of the protocol by presenting a new simulation technique resulting in a concurrent zero-knowledge proof for all languages in NP with a polylogarithmic number of rounds. The analysis of the Kilian-Petrank simulator has been further improved by Prabhakaran and Sahai [37] 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 $\text{COMMIT}(v_i, r_i) = c_i$ 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 [6] 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 [38, 32, 37], Kilian, Petrank, Prabhakaran, Richardson and Sahai show that if the number of rounds in the preamble is set to any superlogarithmic function $\omega(\log k)$ of 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 Our advantage

Our protocol is based on the protocol of [38, 32, 37]. In particular, we use a similar structure of proof system with an important modification. Our proof of correctness relies on the proof in [32] 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 for the commitments. This provides an interesting application of zero-knowledge proofs where the simulator is not only used in the proof of security of the application, but it is invoked by the actual protocol. So, the efficiency of the application directly depends on the efficiency of the simulator.

The main differences between our proof system and the proof systems in [38, 32, 37] 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 [21]). 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 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 [21] used by [32, 37].) 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.

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 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 perfectly binding and computational secrecy properties. These commitments will be later used by the prover of the concurrent zero-knowledge proof. The verifier may use standard commitment schemes without the extra properties. (To allow the zero-knowledge proof to be interactive proof and not just an argument, the verifier commitment scheme should be perfectly hiding).

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 $c = \text{COMMIT}_r(v)$. Informally this proof system has the following 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).
- **Soundness and 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 $c = \text{COMMIT}_r(v) = 1$.

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 $c = \text{COMMIT}_r(v)$ 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 4.1 *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 $c = \text{COMMIT}_r(v)$ (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 $c = \text{COMMIT}(v')$ (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 $\text{COMMIT}_r(v) = c$, 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 [12, 31].

We will now show how to obtain simulatable commitment schemes (in section 5 below), and then proceed with using simulatable commitment to implement efficient concurrent zero knowledge proof systems (see Section 6 below).

5 Commitment schemes based on DDH

In this section we construct a simulatable commitment scheme and prove its properties under the DDH assumption. Our scheme is based on exponentiation in finite groups, but it is quite different from other discrete logarithm based commitment schemes, like Pedersen's [36]. In fact, Pedersen's algorithm is statistically hiding, and only computationally binding. So, it cannot be used in our application, where the committer (the prover) is not necessarily computationally bounded. More importantly, what distinguishes the commitment scheme presented in this section from other standard commitment schemes, is the simulatability property, as defined in the previous section. In particular, we show that our commitment scheme admits a very efficient zero knowledge proof system with perfect completeness and optimal soundness, with 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.

The commitment scheme uses the following parameters: two sufficiently large primes P, Q such that Q divides $P - 1$, and two distinct generators g, h of the order Q subgroup of Z_P. Such parameters are standard in modern cryptography, but for self containment let us say a few words on how they can be chosen. One may consider primes P of the form $2Q+1$ (a so called Sophie-Germain prime, also known in the cryptographic community as a safe prime), however we only need Q to be sufficiently big. This parameters can be chosen by the prover at random, and the verifier simply checks that P, Q are prime, Q divides $P-1$, and Q is sufficiently large. Regarding the generators, the prover picks g', h' uniformly at random in Z_P, computes $g = (g')^{(P-1)/Q} \pmod P$ and $h = (h')^{(P-1)/Q} \pmod P$, and checks that $g, h \neq 0, 1$. It follows from elementary group theory that g and h have (multiplicative) order Q and therefore each of them is a generator of the subgroup of Z_P^* of order Q. The verifier can also check that g and h are generators of the order Q subgroup of Z_P^*, computing g^Q and h^Q (modulo P), and checking that they evaluate to 1.

Then, in order to commit to a value $v \in \{0, \ldots, Q-1\}$, the committer picks uniformly at random $r \in \{0, \ldots, Q-1\}$, and outputs $(P, Q, g, h, g^r, h^{r+v})$. The exponentiations are done modulo the prime P. We will not explicitly say that, but we assume working in Z_P^* for the rest of this paper. There is no need to send (P, Q, g, h) over and over, and the same parameters can be used for several commitments. Upon receiving the commitment $(\bar{g}, \bar{h}) = (g^r, h^{h+r})$, the verifier also checks that $\bar{g}^Q = \bar{h}^Q = 1 \pmod P$, to make sure that both $\bar{g}$ and $\bar{h}$ both belongs to the group generated by g and h. 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. 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.

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 and h generate the subgroup of Z_P^* of order Q, 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 $(g, h, \hat{g}, \hat{h})$ is a commitment to v, i.e., that $(\hat{g}, \hat{h}) = (g^r, h^{r+v})$ for some r. The common input to the proof is a tuple $(P, Q, g, h, \hat{g}, \hat{h})$. The prover gets as an auxiliary input the value r and the proof proceeds as follows.

- The prover selects uniformly at random an $s \in \{0, \ldots, Q-1\}$ and sends $(\bar{g}, \bar{h}) = (g^s, h^s)$ to the verifier.
- The verifier chooses uniformly at random a $q \in \{0, \ldots, Q-1\}$ and sends it to the prover.
- The prover sends $a = qr + s$ to the verifier
- The verifier checks that $\hat{g}^q \bar{g} = g^a$ and $(\hat{h}/h^v)^q \bar{h} = h^a$.

The above proof system is clearly a 3-round public coin proof. 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. Also, the prover can run in polynomial time given r.

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 the order Q subgroup of Z_P^* (this follows from the check $\hat{g}^Q = \hat{h}^Q = 1$ performed by the verifier upon receiving the commitment), and g and h are both generators for this subgroup, 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

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

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 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), as required in Definition 2.1. Note that the second requirement in Definition 2.1, i.e., that the simulator does not rewind the verifier, also holds.

6 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.

6.1 An overview

We modify the concurrent zero-knowledge proof system of [38, 32, 37] 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 [38, 32, 37], 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 [38, 32, 37], 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 toss 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 of setting 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) 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.

6.2 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 protocol. Without loss of generality we assume that the verifier messages are chosen from a set $\mathcal{Q}$.[1] 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 $0 \leq i \leq k$) 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] Shorter messages can be emulated by letting the prover ignore part of the message, while longer messages can be emulated by concatenating several blocks.

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 $c_i = \text{COMMIT}_r(v_i')$ 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_i, v_i) to obtain transcript $(m_{i,j}, q_{i,j}, a_{i,j})$. Then it computes $p_i = P_L(q_1, \ldots, q_{i-1})$ 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 $q_{i,1}, \ldots, q_{i,\ell}$ and $a_{i,1}, \ldots, a_{i,\ell}$ to the verifier.
   (d) The verifier checks that $V_{com}(m_{i,j}, q_{i,j}, a_{i,j}) = \textit{accept}$ for all $j = 1, \ldots, \ell$ and computes $q_i = \bigoplus_j q_{i,j} \oplus q_i'$.
4. The verifier accepts if and only if V_L accepts $(p_1, q_1, \ldots, p_k)$.

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 6.1 *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 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.*

- *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.*
- *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.*
- *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.*

In the rest of this section we sketch the proof of Theorem 6.1.

Completeness Here both the prover and verifier are honest. Since the proof system (P_{com}, V_{com}) for the commitment scheme has perfect completeness and the simulator S_{com} produces a distribution which is polynomial time indistinguishable from the real interaction (actually it is identical to the real interaction for our construction of simulatable commitment scheme), the verifier reaches the last step of the protocol except with negligible probability and the only condition to be checked is that V_L accepts $(p_0, \ldots, p_k)$. It is easy to see that if the q_i' 's are chosen independently and uniformly at random and if the prover is honest, then also the q_i 's are chosen independently and uniformly at random. Also, the prover messages (the p_i's) are computed according to the original protocol P_L. Therefore, the probability that the verifier accepts x is almost the same as in the original proof system, with a negligible difference. In fact, using our specific construction of a simulatable commitment scheme that achieves perfect simulation and perfect completeness, $\epsilon_1 = 0$ and the completeness of the new scheme is exactly err_c.

Soundness Since the verifier uses a perfectly (or statistically) hiding commitment scheme to commit to v_i, the prover has no information about v_i before sending the commitment c_i to the verifier. So, the probability that c_i is a valid commitment to v_i is exactly $1/|V|$ (plus a negligible term if the verifier's commitment scheme is only statistically hiding). It follows that (c_i, v_i) is not a valid pair for all i with probability at least $1 - \ell/|V|$. We now show that, given that all pairs (c_i, v_i) are not valid, the conditional soundness error is the same as in the original protocol. Assume that none of (c_i, v_i) is valid. It follows from the optimal soundness property of (P_{com}, V_{com}) that for any $m_{i,j}$ there exists at most a single $q_{i,j}$ such that $V_{com}(m_{i,j}, q_{i,j}, a_{i,j})$ accepts. Assume that one such value exists (otherwise, the verifier certainly rejects the proof). Then, assuming that q'_i is chosen uniformly at random, the conditional distribution of q_i (given all $m_{i,j}$) is also uniformly random. This proves that the values q_i are chosen as prescribed by the honest verifier V_L, and the probability that V_L accept x (given that all $v_i \neq v'_i$) is at most err_s. Overall, the soundness error is at most $err_s + \epsilon_2$ where $\epsilon_2 = \ell/|V|$.

6.3 Concurrent zero-knowledge

Let us describe a simulator to simulate the above proof system. The simulator gets black box access to V^* – an adversarial verifier that controls a polynomial number of concurrent sessions as discussed in Section 2.2. The simulator will use the following polynomial time machines as subroutines:

1. The simulator S_L for the honest verifier in the original proof system for L.
2. The prover (P^0_{com}, P^1_{com}) of the simulatable commitment scheme.
3. The simulator S_{com} of the simulatable commitment scheme.

Recall that we denote Steps (1) and (2) of the protocol the *preamble* of the proof. Step (3) is the *body* of the proof. Step (2) of the protocol contains ℓ pairs of rounds. Since the simulator has black-box access to the verifier, it may rewind the verifier after it reveals the value v_i, return to just before committing to v'_i and set $v'_i = v_i$. By doing that, it gets extra power for the body of the proof, to be discussed later. A problem with such a rewinding in the setting of concurrent zero-knowledge is that rewinding one step in one proof session may render irrelevant the simulation of steps in other sessions that took place in between those steps. Using the rewinding schedule of [32], [37] shows that if the number of pairs of rounds in the preamble is $\ell(n) = \omega(\log n)$ then the rewinding schedule of [32] has the following two properties. First, it can be run in polynomial time. Second, when run, except for a negligible probability, the simulator gets the chance to set $v'_i = v_i$ for some i and for all of the sessions before getting to simulate the body of the proofs. Namely, for each session, there exists a pair of rounds i, $1 \leq i \leq \ell$ such that the simulator gets a chance to commit to v'_i after it has already seen the verifier revealing the value of v_i. In what follows we assume the above. Note that the transcripts of the preambles that are simulated in the above manner are indistinguishable from transcripts of the real interaction. Of course, the transcript does not show the rewinds, but also, the fact that there exists a round in which the prover has committed on the value $v'_i = v_i$ cannot be told in polynomial time because the commitment scheme of the prover is computationally hiding and v'_i is never revealed. The rewinding schedule of [32] assumes that the simulator does not rewind the verifier during the simulation of the body of the proof. This indeed will be the case here. It is also important to note that the values v'_i (for $1 \leq i \leq \ell$) are never revealed later in the proof. To sum up, until now, we have used the simulation technique and analysis of [32, 37] to get for each session an index i for which $v'_i = v_i$ in polynomial time and such that the preamble transcripts are indistinguishable from the transcripts of the real proof.

We go on with describing how the simulator simulates the bodies of the proofs, i.e., Step (3), without rewinding the verifier V^*. We concentrate on a specific proof session. The same algorithm is used with any of the proof sessions. Let the index in which the simulator has $v_i = v'_i$ be u. The simulator keeps a data structure in which the current session has the entry (u, c_u, v_u, r_u), where $c_u = \text{COMMIT}_{r_u}(v_u)$ is the prover commitment on $v_u = v'_u$ in the preamble.

The simulator runs the honest verifier simulator S_L from the original protocol to produce a sequence of messages $(p_1, q_1, \ldots, p_k)$. The simulator is going to force these p_i's and q_i's to actually appear in Step (3) of the proof, no matter what the cheating verifier V^* (given as a black box) does. Unless, of course, V^* chooses to abort, in which case the original prover would choose to abort as well, and so will our simulator. Step (3)

of the protocol is now executed with the simulator playing the prover, and the adversary V^* generating the responses of the verifiers. The simulator plays the prover in Step (3) in the following manner.

- In step 3.a, when $j = u$, instead of running the simulator S_{com} to obtain $(m_{i,u}, q_{i,u}, a_{i,u})$, run P^0_{com} on input (c_u, v_u, r_u) to obtain message $m_{i,j}$ and state s.
- Instead of setting $p_i = P_L(q_1, \ldots, q_{i-1})$, use the p_i produced by S_L.
- Let q'_i be the message received by the (possibly cheating) verifier. Set $q_{i,u} = \bigoplus_{j \neq u} q_{i,j} \oplus q_i \oplus q'_i$ and compute $a_{i,u} = P^1_{com}(s, q_{i,u})$.

6.3.1 Indistinguishability

As discussed above, the simulated preambles are indistinguishable from the real interaction preambles. We go on with discussing the bodies of the proofs. Hybrid arguments can be used to show that the full proof is indistinguishable as a whole. We note that the hybrid arguments are made possible by the fact that the original prover is auxiliary input zero-knowledge and by the fact that the original prover can run in polynomial time given a witness to x being in L. We do not elaborate on the hybrid composition here. A similar argument can be found in [16]. Loosely speaking, since the verifiers will not be rewound during the bodies of the proofs, concurrent sessions do not interfere during the bodies of the proofs, and we may consider the indistinguishability of each session separately.

Since we are proving zero-knowledge, we may assume that the prover is acting according to the protocol. The honest prover always invokes the simulator S_{com} on pairs (c, v) where c is a commitment to a uniformly chosen random element in $\mathcal{Q}$. By Claim 4.1, the output of the simulator S_{com} in this case is indistinguishable from the output of the simulator S_{com} on inputs (c, v) in which c is a commitment to v. It follows that (except for a negligible probability) the simulator S_{com} must output triplets of the form (m, q, a) and furthermore, the distribution of q in these triplets is indistinguishable from a uniform choice in $\mathcal{Q}$ (as is the case with the output of the simulator S_{com} on legitimate inputs). Thus, we get that the each of the values $q_{i,j}$ for any $1 \leq i \leq k$ and $1 \leq j \leq \ell$ is drawn from a pseudo-random distribution on $\mathcal{Q}$. (In fact, using our DDH based simulatable commitment, the distribution is uniformly random over Q.)

We go on and claim that the verifier V^*, upon seeing a value $m_{i,j}$, generated by the simulator on an illegitimate input pair (c, v), cannot "tell" anything about $q_{i,j}$. Even though $q_{i,j}$ is determined by $m_{i,j}$, if the polynomial time adversarial verifier can change its behavior in a way that distinguishes the value of $q_{i,j}$ from a uniformly chosen value in $\mathcal{Q}$, then we get a contradiction to Claim 4.1. Recall that by this claim, it is not possible to tell transcripts output by the simulator on legitimate inputs from transcripts output by the simulator on illegitimate inputs in polynomial time. Thus, if the prover behaves according to the protocol and the verifier outputs any q'_i, the value $q_i = \bigoplus_j q_{i,j} \oplus q'_i$ is pseudo-random in $\mathcal{Q}$. We deduce that the projection of this proof system on the original proof system, i.e., the distribution of the tuple $(p_1, q_1, \ldots, p_k)$ is polynomially indistinguishable from the distribution of the original interaction (P_L, V_L) with the honest verifier. Our simulator starts by invoking S_L to get an interaction $(p_1, q_1, \ldots, p_k)$ of this type with the appropriate distribution. It then has to fill in the rest of the messages: $m_{i,j}, q_{i,j}, a_{i,j}$ for $1 \leq i \leq k$ and $1 \leq j \leq \ell$ and q'_i for $1 \leq i \leq k$ so that they all fit and are indistinguishable from the distribution on these messages in the original interaction.

The easiest to fit with the correct distribution is q'_i. This value is generated by the real verifier V^* given as a black box. For all $j \neq u$, all values $m_{i,j}, q_{i,j}, a_{i,j}$ are generated exactly as in the original interaction. Thus, they perfectly fit the required distribution. Finally, for $j = u$, the real prover of the simulatable commitment scheme is invoked on a legitimate input (c_u, v_u). Of course, the output distribution is indistinguishable from the distribution output by the simulator on the same legitimate inputs. By Claim 4.1 the distribution output by the simulator on legitimate inputs is indistinguishable from the following distribution: choose a random value $v' \in V$, choose a commitment $c = \text{COMMIT}_r(v')$ and invoke the simulator on (c, v_u). But this latter distribution is exactly the distribution output in the real interaction by the real prover: it invokes the simulator on $(\text{COMMIT}_r(v'_u), v_u)$ for v'_u chosen uniformly at random from V. Thus, also the output messages $m_{i,u}, q_{i,u}, a_{i,u}$ are distributed in a way that is indistinguishable from their distribution in the real proof.

Hybrid arguments can now be used to show that all these separate indistinguishability properties imply indistinguishability for the full interaction.

Intuition: Intuitively, the ability to set $v_i = v'_i$ gave the simulator an edge: it could use the prover (P^0_{com}, P^1_{com}) instead of the simulator. Since it can use the prover, the simulator is able to choose $q_{i,u}$ at his will and let the P^1_{com} come up with a matching $a_{i,u}$. Thus, it chooses $q_{i,u}$ in a way that sets q_i at his will. In particular, it sets all the q_i's and p_i's to fit an interaction output by the simulator S_L of the honest verifier proof system. Having managed to set all p_i's and q_i's appropriately, we get that the body of the proof in the simulation is polynomially indistinguishable from the body of the proof in the real interaction.

One can prove, via a standard hybrid argument, that the above distribution is indistinguishable from the one of the real protocol. It follows from [32, 37] that the new protocol is concurrent zero-knowledge.

6.4 Cost of transformed protocol

We now compute the cost of the transformed protocol when using the simulatable commitment schemes presented in Section 5. First, in terms of round complexity, the transformation adds only ℓ rounds: an additive term which is polylogarithmic (in fact, only slightly superlogarithmic) in the security parameter. (Recall that during step (3) of the protocol, parts (c) and (d) are merged with parts (a) and (b) of the following iteration.) An appropriate $\ell(n)$ is any $\ell(n)$ satisfying $\ell(n) = \omega(\log n)$, for example, $\ell(n) = \log n \cdot \log\log n$ will do. The communication complexity of each round (in the body of the proof) is increased by an additive term: $\ell(n)$ many strings. This addition does not apply to the verifier, who sends the same random string in $\mathcal{Q}$ as before.

In terms of computation time for the prover, we first note that each commitment has a cost of two exponentiations. Each invocation of the simulator S_{com} requires four exponentiations. Given a honest verifier zero-knowledge interactive proof with $2k+1$ rounds we need to add $2\ell+1$ preliminary rounds (for the preamble) in which the prover computes ℓ commitments. The prover then needs to run the simulator ℓ times for each of the k random choices the verifier makes in the original proof system. To summarize, the prover needs to add computation of $4k\ell + 2\ell$ exponentiations. The additional computation for the verifier involves the computation of ℓ commitments in the first round, and receiving ℓ commitments from the prover in the remaining rounds of the preamble, for a total of 4ℓ exponentiations. For each of the verifier's rounds the body of the proof the overhead is invoking V_{com} ℓ times, which requires four exponentiations. This sums up to an overall overhead of $4(k+1)\ell$ additional exponentiations to the verifier.

7 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 7.1 *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 6.2 with $\ell = 1$ is an interactive proof for L with the following properties.*

- *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.*

- *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.*

- *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.*

8 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 demonstrate that even generic constructions as the one in [32, 37] that are usually interpreted as mere plausibility results can lead to efficient protocols when properly instantiated.

References

[1] Boaz Barak. How to Go Beyond The Black-Box Simulation Barrier. IEEE, *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, October, 2001.

[2] M. Ben-Or, S. Goldwasser, O. Goldreich, J. Håstad, J. Kilian, S. Micali and P. Rogaway. Everything Provable is Provable in Zero-Knowledge. *Advances in Cryptology — Proceedings of CRYPTO 88,* Lecture Notes in Computer Science 403, Springer-Verlag (1989). S. Goldwasser, ed.

[3] G. Brassard, D. Chaum and C. Crépeau. *Minimum Disclosure Proofs of Knowledge.* In *JCSS*, pages 156–189. 1988.

[4] C. Brassard, C. Crepeau and M. Yung, "Constant-Round Perfect Zero-Knowledge Computationally Convincing Protocols", *Theoretical Computer Science*, Vol. 84, 1991, pp. 23-52.

[5] R. Canetti, O. Goldreich, S. Goldwasser and S. Micali. Resettable zero-knowledge. Revision 1 of Report CR99-042, the *Electronic Colloquium on Computational Complexity (ECCC)* `ftp://ftp.eccc.uni-trier.de/pub/eccc/`, June 2000. A preliminary version appears in *Proc. 32nd Annual ACM Symposium on Theory of Computing* May 2000.

[6] R. Canetti, J. Kilian, E. Petrank, and A. Rosen. Black-Box Concurrent Zero-Knowldege Requires $\tilde{\Omega}(\log n)$ Rounds. *Thirty-Third Annual ACM Symposium on the Theory of Computing*, July 6-8, 2001.

[7] D. Chaum, I. Damgard and J. van de Graaf. Multiparty Computations Ensuring Secrecy of each Party's Input and Correctness of the Output. Proc. of *Crypto 87*, pp. 462.

[8] T. Cohen, J. Kilian and E. Petrank. Responsive Round Complexity and Concurrent Zero-Knowledge. Proceedings of *Advances in Cryptology - ASIACRYPT 2001 7th International Conference on the Theory and Application of Cryptology and Information Security*, Australia, December 2001.

[9] G. Di Crescenzo and R. Ostrovsky. "On Concurrent Zero-Knowledge with Pre-Processing". *Proceedings of Advances in Cryptology (CRYPT0-99)*, pp. 485-502, Springer-Verlag Lecture Notes in Computer Science, Vol 1666. 1999.

[10] I. Dåmgard. "Efficient Concurrent Zero-Knowledge in the Auxiliary String Model." *Advances in Cryptology – Eurocrypt 2000 Proceedings*, Lecture Notes in Computer Science, Berlin: Springer-Verlag, 2000.

[11] I. Dåmgard, T. Pedersen and B. Pfitzmann. On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures. Advances in Cryptology – CRYPTO '93 Proceedings, pp. 250-265. Lecture Notes in Computer Science #773, Berlin: Springer-Verlag, 1994.

[12] A. De Santis, G. Di Crescenzo, G. Persiano and M. Yung. On Monotone Formula Closure of SZK. In *IEEE Symposium on Foundations of Computer Science*, pp. 454-465, 1994.

[13] D. Dolev, C. Dwork, and M. Naor. "Non-malleable cryptography". In *Proceedings of the 23rd Symposium on Theory of Computing*, ACM STOC, 1991.

[14] C. Dwork, M. Naor and A. Sahai. Concurrent Zero-Knowledge. Proceedings,30*th Symposium on Theory of Computing*, pp. 409–428, 1998.

[15] C. Dwork and A. Sahai. Concurrent Zero-Knowledge: Reducing the Need for Timing Constraints. Proceedings, *Advances in Cryptology – Crypto '98.*

[16] U. Feige. Ph.D. thesis, Weizmann Institute of Science, 1990.

[17] U. Feige, D. Lapidot and A. Shamir. Multiple non-interactive zero-knowledge proofs based on a singe random string. In *Proceedings of the 31st Annual IEEE Symposium on the Foundations of Computer Science*, pages 308–317, 1990.

[18] U. Feige and A. Shamir, "Zero Knowledge Proofs of Knowledge in Two Rounds", *Advances in Cryptology – Crypto 89 proceedings*, pp. 526-544, 1990.

[19] U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In Baruch Awerbuch, editor, *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing*, pages 416–426, Baltimore, MY, May 1990. ACM Press.

[20] O. Goldreich. Foundation of Cryptography - Basic Tools. Cambridge Unievrsity Press. 2001.

[21] O. Goldreich and A. Kahan, "How to Construct Constant-Round Zero-Knowledge Proof Systems for NP", *Journal of Cryptology*, Vol. 9, No. 2, 1996, pp. 167–189.

[22] O. Goldreich, H. Krawczyk. On the Composition of Zero-Knowledge Proof Systems. SIAM J. on Computing, Vol. 25, No.1, pp. 169-192, 1996

[23] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that Yield Nothing But their Validity or All Languages in NP Have Zero-Knowledge proof Systems", *Jour. of ACM.*, Vol. 38, 1991, pp. 691–729.

[24] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, Winter 1994.

[25] O. Goldreich, A. Sahai, and S. Vadhan. Honest-verifier statistical zero-knowledge equals general statistical zeroknowledge. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*, pages 399–408, 1998.

[26] S. Goldwasser, S. Micali, C. Rackoff. The Knowledge Complexity of Interactive Proofs. Proc. 17th STOC, 1985, pp. 291-304.

[27] S. Goldwasser, S. Micali, and C. Rackoff. "The Knowledge Complexity of Interactive Proof Systems", *SIAM J. Comput.*, 18 (1):186–208, 1989.

[28] R. Impagliazzo and M. Yung. Direct Minimum-Knowledge computations. *Advances in Cryptology — Proceedings of CRYPTO 87,* Lecture Notes in Computer Science 293, Springer-Verlag (1987).

[29] J. Kilian. Zero-Knowledge with Log-Space Verifiers
Proceedings, 29th annual IEEE Symposium on the Foundations of Computer Science.

[30] J. Kilian. Achieving zero-knowledge robustly. In A. J. Menezes and S. A. Vanstone, editors, *Advances in Cryptology – CRYPTO '90*, volume 537 of *Lecture Notes in Computer Science*, pages 313-325, 11-15 August 1990. Springer-Verlag, 1991.

[31] J. Kilian and E. Petrank. An Efficient Non-Interactive Zero-Knowledge Proof System for NP with General Assumptions. *Journal of Cryptography*, Vol. 11, Pages 1–27, 1998.

[32] J. Kilian and E. Petrank. Concurrent zero-knowledge in poly-logarithmic rounds. In *Proceedings of the thirty third annual ACM Symposium on Theory of Computing.* ACM Press, 2001.

[33] J. Kilian, E. Petrank, and C. Rackoff. "Lower Bounds for Zero-Knowledge on the Internet", *Proceedings of the 39nd IEEE Conference on the Foundations of Computer Science*, November 1998.

[34] M. Naor. "Bit Commitment Using Pseudo-Randomness,", *Journal of Cryptology*, vol. 4, 1991, pp.151-158.

[35] Y. Oren. On the cunning powers of cheating verifiers: Some observations about zero knowledge proofs. In Ashok K. Chandra, editor, *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 462–471, Los Angeles, CA, October 1987. IEEE Computer Society Press.

[36] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Proc. CRYPTO 91, pages 129-140. Springer-Verlag, 1991.

[37] M. Prabhakaran and A. Sahai. Concurrent Zero Knowledge Proofs with Logarithmic Round-Complexity. Electronic Colloqium on Computational Complexity, ECCC, 2002.

[38] Ransom Richardson and Joe Kilian. On the Concurrent Composition of Zero-Knowledge Proofs. In Proceeedings of *Advances in Cryptology - EUROCRYPT '99*, May 1999, Lecture Notes in Computer Science Vol. 1592 Springer 1999, pp. 415-431

[39] A. Rosen. A Note on the Round-Complexity of Concurrent Zero-Knowledge. *Advances in Cryptology – Crypto 2000 proceedings*, Lecture Notes in Computer Science 1880, p. 451

A Basic Definitions

In this appendix we recall some fundamental definitions concerning zero knowledge proofs. A function $f(n)$ is negligible if for all $c > 0$ and all sufficiently large n, $|f(n)| < 1/n^c$.

A.1 Zero-knowledge proofs

Let us recall the concept of interactive proofs, as presented by [27]. For formal definitions and motivating discussions the reader is referred to [27].

Definition A.1 *A protocol between a (computationally unbounded)* prover P *and a (probabilistic polynomial-time)* verifier V *constitutes an* interactive proof *for a language L if there exist two negligible functions ε_s and ε_c such that*

- **Completeness:** *If $x \in L$ then*

$$\Pr\left[(P,V)(x) \text{ accepts}\right] \geq 1 - \varepsilon_c(|x|)$$

- **Soundness:** *If $x \notin L$ then for any prover P^**

$$\Pr[(P^*, V)(x) \text{ accepts}] \leq \varepsilon_s(|x|)$$

The error in the first item, ε_c is called the completeness error and the error in the second item, ε_s, is called the soundness error.

Brassard, Chaum, and Crépeau [3] suggested a modification of interactive proofs called *arguments* in which the prover is restricted to run in polynomial time. Thus, the soundness property is modified to be guaranteed only for probabilistic polynomial time provers P^*.

Let $(P, V)(x)$ denote the random variable that represents V's view of the interaction with P on common input x. The view contains the verifier's random tape as well as the sequence of messages exchanged between the parties.

We briefly recall the definition of black-box zero-knowledge [27, 35, 22, 26]. The reader is referred to [26] for more details and motivations.

Definition A.2 *A protocol (P, V) is* computational zero-knowledge *(resp.,* statistical zero-knowledge*) over a language L, if there exists an polynomial time oracle machine S (simulator) such that for any polynomial time verifier V^* and for every $x \in L$, the distribution of the random variable $S^{V^*}(x)$ is polynomially indistinguishable from the distribution of the random variable $(P, V^*)(x)$ (resp., the statistical difference between $M(x)$ and $(P, V)(x)$ is a negligible function of $|x|$).*

In this paper, we concentrate on computational zero-knowledge. In the sequel we will say *zero-knowledge* meaning *computational zero-knowledge.*

A (seemingly) weaker definition is one that requires that a simulation exists only for the honest verifier.

Definition A.3 *A protocol (P, V) is* honest-verifier zero-knowledge *over a language L, if there exists a polynomial time machine S (simulator) such that for for every $x \in L$, the distribution of the random variable $S(x)$ is polynomially indistinguishable from the distribution of the random variable $(P, V)(x)$.*

A.2 Auxiliary Input Zero-Knowledge

Auxiliary input zero-knowledge proofs were presented by [27]. Introducing auxiliary input into zero-knowledge enabled sequential composition of zero-knowledge proofs.

Definition A.4 *A proof system (P, V) is* auxiliary-input *zero-knowledge for a language L, if for any $V'(x, y)$ which is polynomial time in the length of its first input, there exists a simulator S', running in time polynomial in the length of its first input, such that the following two distributions are polynomially indistinguishable over the inputs in the language L.*

1. *The view of the verifier $V'(x, y)$ in the interaction with the prover on common input $x \in L$, while it has private access to a string $y \in \{0,1\}^*$ as an auxiliary input.*
2. *The output of the simulator S given (x, y) as the input.*

A.3 Black box simulation

The initial definition of zero-knowledge [26] requires that for any probabilistic polynomial time verifier $\hat{V}$, a simulator $S_{\hat{V}}$ exists that simulates $\hat{V}$'s view. Oren [35] proposes a seemingly stronger, "better behaved" notion of zero-knowledge, known as *black-box* zero-knowledge. The basic idea behind black box zero-knowledge is that instead of having a new simulator $S_{\hat{V}}$ for each possible verifier, we have a single probabilistic polynomial time simulator S that interacts with each possible $\hat{V}$. Furthermore, S is not allowed to examine the internals of $\hat{V}$, but must simply look at $\hat{V}$'s input/output behavior. That is, it can have conversations with $\hat{V}$ and use these conversations to generate a simulation of $\hat{V}$'s view that is computationally indistinguishable from $\hat{V}$'s view of its interaction with P.

At first glance, the limitations on S may seem to force S to be as powerful as a prover. However, S has important advantages over a prover P, allowing it to perform simulations in probabilistic polynomial time.

First, it may set $\hat{V}$'s coin tosses as it wishes, and even run $\hat{V}$ on different sets of coin tosses. More importantly, S may conceptually "back up" $\hat{V}$ to an earlier point in the conversation, and then send different messages. This ability derives from S's control of $\hat{V}$'s coin tosses; since $\hat{V}$ otherwise operates deterministically, S can rerun it from the beginning, exploring different directions of the conversation by trying various messages.

Until recently, essentially all known zero-knowledge proofs were shown zero knowledge via a black-box simulator. Recently, Barak [1] presented a zero knowledge protocol where the simulator uses as input a description of the cheating verifier V^*, and it cannot be implemented given only access to the input/output behavior of V^*.

An advantage of black-box simulator is that they automatically implies zero-knowledge with respect with auxiliary input.

A.4 Witness Indistinguishability

Witness indistinguishable proofs were introduced in [19]. The motivation was to provide a cryptographic mechanism whose notion of security is similar though weaker than zero-knowledge, it is meaningful and useful for cryptographic protocols, and the security is preserved in asynchronous compositions.

We say that a relation R is polynomial time if there exists a machine that given (x, w) works in polynomial time in $|x|$ and determines whether $(x, y) \in R$ or not. For any NP language there exists a polynomial time relation R_L such that L can be described as $L = \{x : \exists y, \quad R_L(x, y)\}$. If $(x, w) \in R$, then w is called a *witness* for $x \in L$. A witness indistinguishable proof is an interactive proof system for a language in NP such that the prover is using some witness to convince the verifier that the input is in the language, yet, the view of the verifier in case the prover uses witness w_1 or witness w_2 is polynomial time indistinguishable. Thus, the verifier gets no knowledge about which witness is used by the prover. The formal definition follows. For further discussion and motivation the reader is referred to [19].

Definition A.5 *A proof system (P, V) is witness indistinguishable with respect to a polynomial time relation R if for any V', any large enough x, any w_1, w_2 such that $(x, w_1) \in R$ and $(x, w_2) \in R$, and for any auxiliary input y for V', the view of V' in the interaction with $P(x, w_1)$ is polynomially indistinguishable from the view of V' in the interaction with $P(x, w_2)$.*

It is shown in [19] that witness indistinguishability is preserved under asynchronous composition of proofs. Moreover, zero-knowledge proofs for languages in NP are also witness indististinguishable. So, any (non concurrent) zero-knowledge proof, like the one in [21], is also a concurrent witness indistinguishable proof system.

B Perfectly hiding commitments

In this section we provide a standard commitment scheme. This is a variant of a scheme, due to Pedersen [36], based on the discrete log assumption (and thus, weaker than the DDH assumption) with perfectly hiding and computationally binding properties. This is the scheme used by the verifier in our interactive proof system. It should be noted that the verifier can use *any* commitment scheme with perfect (or statistical) hiding properties. However, Pedersen's scheme naturally fit our application because it uses parameters that are similar to those used by our simulatable commitment scheme. The only difference between the scheme below, and the one in [36], is in the choice of the parameters P, Q, g, h. In [36] it is assumed that this parameters are either chosen by a trusted center or they are generated by the participating parties using a coin-flipping protocol. In our application, the choice of the parameters can be safely delegated to the prover (i.e., the recipient of the commitments), avoiding the need of a trusted center.

Beside the cost of generating the parameters (which can be used for several commitments), the cost of each commitment scheme is two exponentiations. The setting is similar to our simulatable commitments. Namely, the prover (who plays the role of the receiver this time) choose two sufficiently large primes P, Q such that Q divides $P-1$, and two random distinct generators g, h of the order Q subgroup of Z_P. This can be achieved selecting g', h' in Z_P^* at random, computing $g = (g')^{(P-1)/Q}$, $h = (h')^{(P-1)/Q}$, and checking that $g, h \neq 0, 1$. The receiver sends these parameters to the committer (verifier), who checks that P, Q are primes,

Q divides $P-1$, and g and h have order Q. The committer may use parameters P, Q, g, h several times. In fact, the same parameters can be used also for the perfectly binding simulatable commitment scheme used by the prover.

To commit to a value $v \in \{0, \ldots, Q-1\}$, the committer chooses a random $r \in \{0, \ldots, Q-1\}$ and sends the value $g^r \cdot h^v \pmod{P}$ to the receiver.

Perfect secrecy: Whatever v is, if r is chosen uniformly at random in $\{0, \ldots, Q-1\}$ (and g is a generator of the order Q subgroup of Z_P), then the commitment $g^r \cdot h^v \pmod{P}$ is a uniformly chosen value in the order Q subgroup of Z_P.

Computational binding: If the verifier may come up with (r_1, v_1) and (r_2, v_2) such that

$$g^{r_1} \cdot h^{v_1} = g^{r_2} \cdot h^{v_2} \pmod{P},$$

for $v_1 \neq v_2 \pmod{Q}$, then the verifier may also compute the discrete log of h to the base of g as $\log_g h = (r_1 - r_2)/(v_2 - v_1) \pmod{Q}$.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'