Concurrent Statistical Zero-Knowledge Arguments for NP from One Way Functions

Vipul Goyal[⋆], Ryan Moriarty[⋆⋆], Rafail Ostrovsky[⋆⋆⋆], and Amit Sahai[†]

University of California, Los Angeles
{vipul, ryan, rafail, sahai}@cs.ucla.edu

Abstract. In this paper we show a general transformation from any honest verifier statistical zero-knowledge argument to a concurrent statistical zero-knowledge argument. Our transformation relies only on the existence of one-way functions. It is known that the existence of zero-knowledge systems for any non-trivial language implies one way functions. Hence our transformation *unconditionally* shows that concurrent statistical zero-knowledge arguments for a non-trivial language exist if and only if standalone secure statistical zero-knowledge arguments for that language exist.

Further, applying our transformation to the recent statistical zero-knowledge argument system of Nguyen et al (STOC'06) yields the first concurrent statistical zero-knowledge argument system for all languages in **NP** from any one way function.

1 Introduction

Zero-knowledge proof systems were introduced by Goldwasser, Micali and Rackoff [GMR89] and have the remarkable property that they yield nothing except the validity of assertion being proved. Such protocols involve a prover, who tries to prove some assertion, and a verifier, who is trying to decide if he believes the assertion. A cheating prover may act maliciously by trying to prove a false statement; a cheating verifier may try to learn more than the validity of the statement being proved. The property that the verifier learns nothing (except the validity of the statement) is formalized as the *zero-knowledge* condition and

⋆ Research partially done while visiting IPAM. Supported in part by grants listed below.

⋆⋆ Research partially done while visiting IPAM. Supported in part by grants listed below.

⋆⋆⋆ Research partially done while visiting IPAM. This research was supported in part by IBM Faculty Award, Xerox Innovation Group Award, NSF Cybertrust grant no. 0430254, and U.C. MICRO grant.

† Research partially done while visiting IPAM. This research was supported in part by NSF ITR and Cybertrust programs (including grants 0627781, 0456717, and 0205594), a subgrant from SRI as part of the Army Cyber-TA program, an equipment grant from Intel, and an Alfred P. Sloan Foundation Research Fellowship.

the property that the prover cannot prove a false statement is formalized as the *soundness* condition.

Depending upon how strong we want the zero-knowledge property or the soundness property to be, we can define several different types of zero-knowledge systems. In *statistical zero-knowledge*, we require the zero-knowledge condition to hold even against an infinitely powerful cheating verifier. When we relax the zero-knowledge condition so that it need only hold against a probabilistic polynomial time cheating verifier, we get the so called *computational zero-knowledge.* Similarly, we can have zero-knowledge with either *statistical soundness* (known as zero-knowledge *proof systems*) or just *computational soundness* (known as zero-knowledge *argument systems*).

It would be desirable to construct statistical zero-knowledge proof systems for all languages in **NP**. Unfortunately it was shown that such systems can only be obtained for languages in **AM**$\cap$**coAM** [BHZ87], and **AM**$\cap$**coAM** cannot contain **NP** unless the polynomial hierarchy collapses. Thus if we want a zero-knowledge system for all language in **NP**, we can only have either statistical soundness or statistical zero-knowledge (but not both).

The original definition of zero-knowledge considers protocols running alone in isolation. That is, we have a single prover interacting with a single verifier. The concurrent setting was introduced by Dwork et al [DNS98] (see also [Fei90]) with a motivation to construct zero-knowledge protocols for more realistic settings (such as when the protocols are to be executed over the Internet). In the concurrent setting, many protocol executions are run at the same time with possibly a single prover simultaneously talking to many verifiers. The prover in this setting runs the risk of a coordinated attack from many different verifiers which interleave the execution of protocols and choose their responses to the prover based on each others' messages. If a zero-knowledge protocol maintains its zero-knowledge property even in the concurrent setting, it is said to be *concurrent zero-knowledge.*

Our Results. We give the first general transformation from any zero-knowledge system to concurrent zero-knowledge system that maintains the statistical zero-knowledge property of the system. Hence our compiler can be used to transform a computational zero-knowledge argument system into a concurrent computational zero-knowledge argument system as well as a statistical zero-knowledge argument system into a concurrent statistical zero-knowledge argument system. Our transformation only relies on the existence of one-way functions. Further, it does not require that the original protocol be public coin. These properties separate it from the compiler in [MP03], since the compiler in [MP03] was designed to maintain statistical soundness (whereas we deal with statistical zero-knowledge) and was designed to be very efficient (our transformation is polynomial time but we do not optimize for efficiency). Additionally, the compiler in [MP03] relies on specific number theoretic assumptions.

We would like to emphasize that our compiler only uses one-way functions. It is known that the existence of zero-knowledge systems for any non-trivial language implies one way functions [OW93]. Hence our transformation *unconditional-*

ally shows that concurrent statistical zero-knowledge arguments for a non-trivial language exist if and only if standalone secure statistical zero-knowledge arguments for that language exist. This feature also allows us to achieve a main goal of ours: applying our transformation to the statistical zero-knowledge system from [NOV06], we get the first concurrent statistical zero-knowledge argument system for an **NP**-complete language from any one-way function.

Techniques. Here we describe our techniques at a high level. Our goal is to create a general compiler that will work for *honest verifier* statistical zero-knowledge arguments and turn them into concurrent statistical zero-knowledge arguments. We first use a modified version of the preamble from the concurrent zero knowledge protocol of [PRS02]. Using a preamble similar to [PRS02] enables us to have a verifier committed to his randomness for the run of the protocol and to give a strategy for a simulator that could extract that randomness in the concurrent setting. Thus we are be able to use a straight-line simulator after the preamble.

The main technical challenges are to adapt the preamble of [PRS02] to work with an all-powerful verifier and to base the preamble solely on one-way functions. The proof of soundness in [PRS02] relies on the verifier using statistically hiding commitments to commit to its randomness. However using statistically hiding commitments during the preamble does not seem plausible in our setting even though (independent of this work) they have recently been constructed from one way functions [HR07]. The main reason is that since we are dealing with statistical zero-knowledge, the verifier could potentially be all powerful. Thus all the commitments by the verifier to the prover should be statistically *binding*. Consequently, if the randomness of the verifier is not statistically hidden from the prover during the PRS preamble, it remains unclear how the proof of soundness would go through (even if the *prover* uses statistically hiding commitments).

To overcome this problem, the verifier commits using statistically binding commitments based on one-way functions as it appears essential in our setting. However, the verifier never actually opens the commitment. Instead the verifier gives a (standalone secure computational) zero-knowledge proof that his message are consistent with the randomness committed to in the PRS preamble. Note that it is important that we use a zero-knowledge *proof* here since the verifier is all powerful. This idea enables us to prove that our transformation preserves the soundness of the underlying proof system.

Furthermore, since we are transforming from an honest verifier statistical zero-knowledge argument into a concurrent statistical zero-knowledge argument, we need to find a way to relax the requirement that the verifier is honest. In order to achieve this goal, the randomness that the verifier uses is determined by a coin-flipping protocol between the prover and the verifier (instead of being chosen freely by the verifier alone). This is important for our proof of the zero-knowledge condition since our simulator for the underlying protocol will require verifier responses with correctly distributed randomness. Also, this technique combined with the trick of using zero-knowledge proofs from the verifier allows us to deal with *private-coin* protocols as well.

We are able to combine all of these ideas into a single compiler that lets us achieve our results.

1.1 Related Work

Statistical zero-knowledge arguments. In this paper, we will be examining statistical zero-knowledge arguments which were first introduced by [BCC88]. From the constructions of [GMW91, BCC88] it is clear that one main technique to construct statistical zero-knowledge arguments for any language in **NP** is to first construct statistically hiding commitments (and plug them into a standard protocol).

Early constructions of statistically hiding commitments were built on specific number theoretic assumptions [BCC88, BKK90]. In [GK96] it was shown how to construct statistically hiding commitments from claw-free permutations; this was further reduced to any family of collision-resistant hash functions in [NY89].

Naor et al [NOVY98] showed how to construct statistically hiding commitments from one way permutations. In [Ost91, OW93] it was shown that one could build a weak from of one-way functions from statistically hiding commitments. Thus one-way functions would be the minimal assumption needed to create statistically hiding commitments. Until recently, no further progress was made. Haitner et al [HHK$^+$05] showed how to construct statistically hiding commitments from a one-way function that could approximate the pre-image size of points in the range.

In a recent breakthrough work, Nguyen et al [NOV06] were able to construct statistical zero-knowledge arguments from any one-way function for all languages in **NP**. They deviated from the traditional line of constructing statistically binding commitments from one way functions. Instead they created a relaxed variant of statistically binding commitments from one-way functions first introduced by Nguyen and Vadhan [NV06]. Building on [NOV06], Haitner and Reingold [HR07] recently constructed statistically hiding commitments from one way functions. We remark that [NOV06] serves as a critical component for our results.

Concurrent zero-knowledge. The notion of concurrent zero knowledge was introduced by [DNS98] (see also [Fei90]) who also gave a construction based on timing assumptions. Richardson and Kilian [RK99] exhibited a family of concurrent zero-knowledge protocols for all languages in **NP** in the plain model. The analysis of the their protocol required that the protocol have a polynomial number of rounds. This analysis was improved by Kilian and Petrank [KP01] who showed that the protocol only required a poly-logarithmic number of rounds. Prabhkaran, Rosen, and Sahai introduced a variant of the protocol and reduced the number of rounds further to $\omega(\log n)$ rounds in [PRS02]. This is the protocol we will mainly use in our general compiler.

In [MP03], Micciancio and Petrank give a general compiler to compile any public-coin honest verifier zero-knowledge proof system into a concurrent zero-knowledge proof system while incurring only an additional $\omega(\log n)$ rounds. This reduction is based on perfectly hiding commitment schemes (having some additional special properties) based on the Decisional Diffie-Hellman assumption. These reductions do not however maintain the statistical zero-knowledge property. In other words, even if the original protocol is statistical zero-knowledge, the resulting protocol may not be.

Concurrent statistical zero-knowledge. There has not been much work on concurrent statistical zero-knowledge. In [MOSV06], Micciancio et al show how to build concurrent statistical zero-knowledge proofs for a variety of problems *unconditionally*, that is, without making any unproven complexity assumptions. However since these were statistical zero-knowledge proofs, their results could not include proofs for all languages in **NP** (unless **NP** is in **AM**∩**coAM** and the polynomial hierarchy collapses).

2 Preliminaries

Statistical Difference The *statistical difference* between two random variables X, Y taking values in a universe $\mathbb{U}$ is defined to be

$$\Delta(X,Y) \stackrel{def}{=} \max_{S\subset\mathbb{U}} \Big|\Pr[X \in S] - \Pr[Y \in S]\Big| = \frac{1}{2}\sum_{x\in\mathbb{U}} \Big|\Pr[X = S] - \Pr[Y = S]\Big|$$

We say two distributions are statistically close if $\Delta(X,Y)$ is negligible.

Definition 1 (Argument Systems ([Gol01])) *An interactive protocol* (P,V) *is an argument (or computationally sound proof system) for a language* L *if the following three conditions hold:*

1. *(Efficiency)* P *and* V *are computable in probabilistic polynomial time.*
2. *(Completeness) If* $x \in L$*, then* V *outputs* accept *with probability at least* $2/3$ *after interacting with the honest prover* P*.*
3. *(Soundness) If* $x \notin L$*, then for every nonuniform PPT adversarial prover* P^**,* V *outputs* accept *with probability at most* $1/3$*.*

For an argument system (P,V), we define the following terms. If $x \in L$, then the value that lower bounds the probability of V outputting accept after interacting with the honest prover P is called the *completeness bound*. Similarly, If $x \notin L$, then the value that upper bounds the probability of V outputting accept after interacting with any nonuniform PPT adversarial prover P^* is called the *soundness error*.

We say that an argument system is public coin if all the messages sent by V are chosen uniformly at random, except for the final *accept/reject* message (which is computed as a deterministic function of the transcript).

Concurrent Zero-knowledge We assume the conversation between the prover P and the verifiers $V_1 \dots V_n$ is of the form $v_1, p_1, v_2, p_2, \dots, v_t, p_t$ where each v_j is a messages sent to the prover from a verifier V_{i_j} and the provers' response is the message p_j. We assume that there is an adversary A which controls the verifiers and the verifiers' messages. The adversary will take as input the partial conversation so far, i.e., $v_1, p_1 \dots v_k, p_k$ and output a pair (i, v) specifying that P will receive message v from verifier V_i. The view of the adversary on input x will include the verifiers' random tapes and all the messages exchanged between the prover and the verifiers. This view will be denoted by $(P, A)(x)$.

Definition 2 *We say that an argument system (P, V) for a language L is statistical (resp., computational) black box* concurrent zero-knowledge *if there exists a probabilistic polynomial time oracle machine S (the simulator) such that for any unbounded (resp., probabilistic polynomial time) adversary A, the distributions $(P, A)(x)$ and $S^A(x)$ are statistically close (resp., computationally indistinguishable) for every string x in L.*

We call the statistical difference of these distributions the *zero-knowledge error* of the protocol. If we are dealing with computational indistinguishability, the probability that a probabilistic polynomial time adversary can distinguish these distributions is called the zero-knowledge error of the protocol as well.

Honest Verifier We say a proof system is an honest verifier proof system if the zero-knowledge property is guaranteed to hold only if the verifier acts according to the protocol.

Note on Notation We will use $P(T, r)$ (resp., $V(T, r)$) to signify the correct next message of an honest P (resp., V) as per the protocol (P, V), given the random coins r and the interaction transcript T observed so far. Sometimes, the random coin r might be implicit (instead of being explicitly supplied as an input).

3 Compiler Parts

In this section, we give the different parts of the compiler in isolation before putting them together in the next section to give our full protocol.

3.1 Underlying zero-knowledge protocol

We assume that as input to our compiler, we have an honest verifier statistical zero-knowledge argument system for some language L. This protocol will have a prover, a verifier, a completeness bound, a soundness error, a simulator, the number of rounds and a zero-knowledge error (denoted by P, V, e_c, e_s, S, t and e_z respectively). We let $p_1, \dots p_t$ denote the messages of the prover and $v_1, \dots v_t$ the messages of the verifier in a particular execution of the argument system.

3.2 Statistically binding commitments from any OWF

In our protocol, we shall use statistically binding commitments from any OWF. Building on techniques from [HILL99], such commitments were constructed by Naor [Nao91].

We denote such a commitment scheme by COM. We denote the probability of an all powerful adversary breaking the binding property of the scheme as b_{com}. We denote the probability of a PPT adversary breaking the hiding property of the scheme as h_{com}.

3.3 Computational zero-knowledge proof based on any OWF for all of *NP*

In our protocol, we shall use a computational zero-knowledge proof based on one-way functions for every language in **NP** with negligible soundness error and perfect completeness. One way to construct them is to create statistically binding commitments based on a OWF as stated earlier [HILL99, Nao91]. These commitments can then be used in the 3-colorability protocol of [GMW91] to give us a zero-knowledge proof for any language in **NP**. We can then repeat the protocol sequentially n^2 times (where n is the security parameter) to achieve negligible soundness error. We note that this protocol will also have perfect completeness. We denote the final protocol after the sequential repetitions as (P', V').

This protocol will have a prover, a verifier, a completeness bound, a statistical soundness error, a simulator, the number of rounds and a zero-knowledge error (denoted by $P', V', e'_c = 1, e'_s, S', t'$ and e'_z respectively).

3.4 Preamble from PRS [PRS02]

In this subsection, we describe the preamble from [PRS02] and give its useful properties for our context. We note that [RK99, KP01] also have similar preambles (with round complexity higher than [PRS02]) which could be used for our purpose.

The preamble of the PRS protocol is simple. Let n be the security parameter of the system and k be any super-logarithmic function in n. Let σ be the bit string we wish to commit to and γ be the length of σ. We break σ up into two random shares k^2 times. Let these shares be denoted by $\{\sigma^0_{i,\ell}\}^k_{i,\ell=1}$ and $\{\sigma^1_{i,\ell}\}^k_{i,\ell=1}$ with $\sigma^0_{i,\ell} \oplus \sigma^1_{i,\ell} = \sigma$ for every i, ℓ. The verifier will commit to these bits using COM with fresh randomness each time. The verifier then sends these k^2 commitments to the prover. This is then followed by k iterations where in the ℓth iteration, the prover sends a random k-bit string $b_\ell = b_{1,\ell}, \ldots, b_{k,\ell}$, and the verifier decommits to the commitments $\mathrm{COM}(\sigma^{b_{1,\ell}}_{1,\ell}), \ldots, \mathrm{COM}(\sigma^{b_{k,\ell}}_{k,\ell})$.

The goal of this protocol is to enable the simulator to be able to rewind and find the value σ with high probability by following a fixed strategy. Since the verifier commitments are set after the first round, once we rewind the verifier, the simulator will have the opportunity to have the verifier open both the σ^0

commitment and the σ^1 commitment. In the concurrent setting, rewinding a protocol can be difficult since one may rewind past the start of some other protocol in the system as observed by [DNS98]. The remarkable property of this protocol is that there is a fixed rewinding strategy the simulator can use to get the value of σ, for every concurrent cheating verifier strategy $\mathbb{V}^*$, with high probability.

We will follow [MOSV06] in formalizing the properties of the PRS preamble we need. Without loss of generality, assume that there are Q concurrent sessions. Recall that k is the number of rounds of the PRS preamble.

We call the simulator for the PRS preamble CEC-Sim. CEC stands for concurrently-extractable commitments. CEC-Sim will have oracle access to $\mathbb{V}^*$ and will get the following inputs.

- Commitments schemes $\mathcal{COM} = COM_1, COM_2, \ldots, COM_Q$, where COM_s is the commitment scheme used for session s.
- Parameters γ, k, n and Q, all given in unary.

We also need to give the following definitions adapted from [MOSV06]:

Definition 3 (Major Decommitment) *A* major decommitment *is a reveal after the PRS preamble in which $\mathbb{V}^*$ reveals the opening of commitments $\{COM(\sigma^0_{i,\ell})\}^k_{i,\ell=1}$ and $\{COM(\sigma^1_{i,\ell})\}^k_{i,\ell=1}$. P only accepts the major decommitment if: (a) all these openings are valid openings to the commitments in the transcript, and, (b) there exists σ such that for all i, ℓ, $\sigma^0_{i,\ell} \oplus \sigma^1_{i,\ell} = \sigma$.*

Definition 4 (Valid Commit Phase) *For a transcript T of the commit phase interaction between P and $\mathbb{V}^*$, let $T[s]$ denote the messages in session s. $T[s]$ is a* valid commit phase transcript *if there exists a major decommitment D such that $P(T[s], D)=$ accept.*

Definition 5 *(Compatibility). Message $M=(\sigma, \sigma^0_{i,j}, \sigma^1_{i,j})$ is* compatible *with $T[s]$ if*

1. *$\sigma = \sigma^0_{i,j} \oplus \sigma^1_{i,j}$*
2. *There exist commitments $COM_s(\sigma^0_{i,j})[s]$ and $COM_s(\sigma^1_{i,j})[s]$ that are part of the transcript of the first message of $T[s]$.*

Observe that if a message $M=(\sigma, \sigma^0_{i,j}, \sigma^1_{i,j})$ is compatible with the transcript $T[s]$, the cheating verifier can major-decommit to a message different from σ only with probability at most b_{com}. Thus we call σ the *extracted message*.

Definition 6 *A Simulator $CEC{-}Sim^{\mathbb{V}^*}$ has the* concurrent extraction *property if for every interaction T it has with $\mathbb{V}^*$, it also provides (on a separate output tape) an array of messages $(M_1, M_2, \ldots, M_Q)$ with the following property:*

For every session $s \in \{1, 2, \ldots, Q\}$, if $T[s]$ is a valid commit phase transcript, then M_s is compatible with $T[s]$.

A simulator that has the concurrently extractable property is also called a *concurrently-extractable simulator*.

Using the simulation and rewinding techniques in [PRS02], we can obtain a concurrently-extractable simulator for the PRS preamble. Let $\langle \mathbb{P}, \mathbb{V}^* \rangle$ denote the output of $\mathbb{V}^*$ after concurrently interacting with $\mathbb{P}$. Recall that $\mathbb{V}^*$ is an unbounded adversary.

Lemma 1. *(implicit in [PRS02], adapted from [MOSV06]). There exists a PPT concurrently-extractable simulator CEC-Sim with a fixed strategy SIMULATE such that for $\mathcal{COM}$ and all concurrent adversaries $\mathbb{V}^*$, for settings of parameters σ=poly(n), $k = \tilde{O}(\log n)$, and Q =poly(n), we have the ensembles*

$$\left\{ \textit{CEC-Sim}^{\mathbb{V}^*}(\mathcal{COM}, 1^\sigma, 1^k, 1^n, 1^Q) \right\}_{n \in \mathbb{N}} \text{ and } \left\{ \langle \mathbb{P}, \mathbb{V}^* \rangle (\mathcal{COM}, 1^\sigma, 1^k, 1^n, 1^Q) \right\}_{n \in \mathbb{N}}$$

have statistical difference ϵ, where ϵ is negligible.

4 The Compiler

In this section, we discuss the compiler in detail. It takes as input an honest verifier statistical zero knowledge argument system (P, V) and compiles it into a concurrent statistical zero knowledge argument system $(\mathbb{P}, \mathbb{V})$ assuming the existence of one way functions. The compiler uses statistically binding commitments and computational zero knowledge proofs as building blocks. Both of these can be constructed out of any one way function [HILL99, GMW91].

The compiler is presented formally in Figure 1. Let R denote the uniform distribution. The verifier $\mathbb{V}$ first generates a random string r (i.e., $r \stackrel{r}{\leftarrow} R$). $\mathbb{P}$ and $\mathbb{V}$ then carry out the PRS preamble [PRS02] where $\mathbb{V}$ sets σ to be r.

Instead of using statistically hiding commitments as in the PRS preamble, we will use statistically binding commitments based on one way functions. This however causes a problem in the PRS soundness proof [PRS02] since the statistical hiding property of the commitments is used in an essential manner in the soundness proof[1]. We resolve this problem later on.

Once $\mathbb{P}$ and $\mathbb{V}$ have finished the PRS preamble,$\mathbb{V}$ gives a computational zero knowledge proof acting as P' in the system (P', V') (constructed using a OWF as described in section 3). It proves that all the shares it committed to in the PRS preamble (first message) are "consistent" with r. In other words, $r^0_{i,\ell} \oplus r^1_{i,\ell} = r$ for every i, ℓ. The prover $\mathbb{P}$ then draws $r' \stackrel{r}{\leftarrow} R$ and sends it to $\mathbb{V}$. Now $\mathbb{P}$ and $\mathbb{V}$ will begin the supplied honest verifier statistical zero knowledge argument protocol (P, V) with some modifications. The random coins of the verifier V are fixed to be $r \oplus r' \stackrel{\text{def}}{=} r''$.

[1] For example, if the verifier uses computationally hiding commitments, a cheating prover could potentially create dependencies between his own commitments and the verifier challenge

Common Input to $\mathbb{P}$ and $\mathbb{V}$: $(P,V),(P',V'),x$,COM
Compiler:

1. $\mathbb{V} \to \mathbb{P}$: Generate $r \xleftarrow{r} R$. Using COM, commit to r and the shares $\{r^0_{i,\ell}\}^k_{i,\ell=1}, \{r^1_{i,\ell}\}^k_{i,\ell=1}$ such that $r^0_{i,\ell} \oplus r^1_{i,\ell} = r$ for every i,ℓ.
2. For $\ell = 1, \ldots k$:
   (a) $\mathbb{P} \to \mathbb{V}$: Send $b_{1,\ell}, \ldots, b_{k,\ell} \xleftarrow{r} \{0,1\}^k$.
   (b) $\mathbb{V} \to \mathbb{P}$: Decommit to $r^{b_{1,\ell}}_{1,\ell}, \ldots, r^{b_{k,\ell}}_{k,\ell}$.
3. $\mathbb{V} \leftrightarrow \mathbb{P}$: Zero-knowledge proof (P',V') where $\mathbb{V}$ acts as P' and proves to $\mathbb{P}$ that $r^0_{i,\ell} \oplus r^1_{i,\ell} = r$ for every i,ℓ and that there exist valid openings to the commitments in the PRS preamble to $r^0_{i,\ell}, r^1_{i,\ell}$. If $\mathbb{P}$ accepts the zero-knowledge proof, the transcript of the commit phase is guaranteed to be a *valid commit phase transcript*.
4. $\mathbb{P} \to \mathbb{V}$: send $r' \xleftarrow{r} R$.
5. $\mathbb{V}$ calculates $r'' \stackrel{\text{def}}{=} r \oplus r'$
6. For $j = 1, \ldots t$:
   (a) $\mathbb{P} \to \mathbb{V}$: send $P(T^P_j) = p_j$.
   (b) $\mathbb{V} \to \mathbb{P}$: send $V(T^V_j, r'') = v_j$.
   (c) $\mathbb{V} \leftrightarrow \mathbb{P}$: zero-knowledge proof (P',V') where $\mathbb{V}$ acts as P' and proves to $\mathbb{P}$ that there exist an r'' such that $r \oplus r' = r''$ and $V(T^V_j, r'') = v_j$.
7. $\mathbb{V} \to \mathbb{P}$: send $V(T, r'') =$ accept/reject.

Fig. 1. Compiler

Let the protocol (P,V) have t rounds where one round involves a prover message followed by the verifier's response. P and V interact as follows. In the jth round, $\mathbb{P}$ calculates the next message p_j of P on the transcript T^P_j of the interaction so far. Transcript T^P_j is defined to contain all the messages exchanged between P and V so far, i.e., $T^P_j = (p_1, v_1, \ldots, p_{j-1}, v_{j-1})$.

The verifier $\mathbb{V}$ receives p_j from $\mathbb{P}$. It will now calculate V's response in the protocol (P,V) using randomness r'' and V's transcript T^V_j $(= (T^P_j, p_j))$ of the interaction so far; we call this response v_j. Now $\mathbb{V}$ will act as the P' in the computational zero-knowledge proof system (P',V').

$\mathbb{V}$ will prove that his response is indeed consistent with V acting on input T^V_j and randomness r''. The statement being proven by $\mathbb{V}$ is in NP since it is possible to check the statement given the opening of the commitment to r. We are using the computational zero-knowledge proof here instead of just revealing the commitments to make our soundness proof go through. $\mathbb{P}$ acts as V' during this zero-knowledge proof. If the proof is accepted by V' then $\mathbb{P}$ accepts v_j.

Once these t rounds are complete, $\mathbb{V}$ accepts if and only if V would accept on the complete transcript T $(=(T^V_t, v_t))$.

4.1 Parameters of the compiler

Let (P,V) be an honest verifier zero-knowledge argument system with t rounds, e_c completeness bound, e_s soundness error, and e_z zero-knowledge error. Let

(P', V') be a computation zero-knowledge proof system with t' rounds, e'_c completeness bound, e'_s soundness error, and e'_z zero-knowledge error. Let ϵ be the value from Lemma 1 that represents the statistical difference of a simulated run of the PRS preamble using SIMULATE from a real run against an arbitrary unbounded concurrent verifier strategy. Let k be the number of rounds in the PRS preamble. Let e_p be the probability that the PRS preamble is accepted by the prover and the verifier if they are behaving honestly. Let COM be the commitment used in the PRS preamble. Let h_{com} be the probability of a PPT machine breaking the hiding property of COM and b_{com} be the probability of an all powerful adversary breaking the binding property of COM. Let S be the simulator for (P, V) and $\mathbb{S}$ be a simulator for $(\mathbb{P}, \mathbb{V})$.

We give the parameters that we obtain with our compiler in the following theorem.

Theorem 1 *Running the compiler given in Section 4 on the argument system (P, V) results in a system $(\mathbb{P}, \mathbb{V})$ with the following properties.*

- *The completeness bound of $(\mathbb{P}, \mathbb{V})$ is $e_p e_c$.*
- *The soundness error of $(\mathbb{P}, \mathbb{V})$ is $e_s + (k^2 h_{com} + e'_z)t$.*
- *The zero-knowledge error of the protocol is:* $\Delta((\mathbb{P}, \mathbb{V}^*)(x), \mathbb{S}^{\mathbb{V}^*(x)}) = \epsilon + e_z + k^2 b_{com} + e'_s t$

Proof. The proof of each of the above claims is given below individually.

Completeness Suppose $x \in L$. Then the probability that the protocol is accepted by $\mathbb{V}$ is:

$$\Pr[(\text{PRS is accepted}) \wedge ((P, V) \text{ is accepted}) \wedge (\text{each execution of } (P', V') \text{ is accepted})] =$$

$$(e_p)(e_c)(e'_c)^t$$

Note that e'_c is one since our protocol (P', V') has perfect correctness. Thus we get the probability that the transformed protocol is accepted is $(e_p)(e_c)$.

Soundness Suppose $x \notin L$ and there exists an adversarial PPT prover $\mathbb{P}^*$ that can get $\mathbb{V}$ to accept with non-negligible probability ϕ. In other words, suppose $(\mathbb{P}, \mathbb{V})$ has non-negligible soundness error ϕ. We will show how to use $\mathbb{P}^*$ to build a machine D that breaks the soundness of the underlying zero-knowledge protocol (P, V). We give a formal description of D in Figure 2.

D will use $\mathbb{P}^*$ as follows. D runs $\mathbb{P}^*$ and executes the PRS preamble interacting with it setting σ to a random r. Now, D gives a computational zero knowledge proof to $\mathbb{P}^*$ and receives r' as shown in Figure 2. It then runs the honest verifier machine V acting a cheating prover P^* and trying to break the soundness of the system (P, V).

In the jth round, D receives p_j from $\mathbb{P}^*$ and sends it to V. V will respond to p_j with v_j. Now D wants to be able to give v_j as his response to $\mathbb{P}^*$ so as to be able to continue the protocol. However D needs his response to $\mathbb{P}^*$ to be generated

Common Input to D and V: x
Auxiliary input to D: The cheating prover machine $\mathbb{P}^*$
Description of D, a cheating prover for (P, V)

1. D runs a copy of $\mathbb{P}^*$, acting as the verifier itself.
2. D generates $r \xleftarrow{r} R$. It then interacts with $\mathbb{P}^*$ to carry out the PRS preamble using r.
3. D gives a zero knowledge proof (P', V') to $\mathbb{P}^*$ proving that all the shares it committed to in the PRS preamble are consistent with r.
4. D receives r' from $\mathbb{P}^*$
5. For $j = 1, \ldots t$:
   (a) D gets the message p_j from $\mathbb{P}^*$.
   (b) $D \rightarrow V$: p_j.
   (c) $V \rightarrow D$: v_j.
   (d) D uses the simulator S' of the system (P', V') and simulates a proof with $\mathbb{P}^*$ that $V(T_j^V, r \oplus r') = v_j$.

Fig. 2. D acting as a cheating prover for (P, V).

using randomness $r \oplus r'$ as per the protocol (P, V). D has already committed to r with a statistically binding commitment and thus can not necessarily decommit to a r such that v_j is consistent with r, r' and (P, V).

However D does not have to decommit to r, but only needs to give a zero-knowledge proof that he has committed to a randomness r such that v_j is consistent with r, r' and (P, V). He can use the simulator of (P', V') to do this. Hence, D sends v_j to $\mathbb{P}^*$ and simulates a zero knowledge proof of its correctness by rewinding $\mathbb{P}^*$. The probability that $\mathbb{P}^*$ can differentiate between such a simulated run and a real run can be analyzed using a simple hybrid argument. As we move from a real run to a simulated one, we construct the following hybrid. D acts as an honest $\mathbb{V}$ sending correct verifier messages v_j. However, instead of giving real zero knowledge proofs, D gives simulated proofs. In other words, although D would have the witness to the NP statement, it does not use it and instead simulates the zero knowledge proof. Clearly, the probability that $\mathbb{P}^*$ can distinguish this hybrid from a real run is bounded by the zero-knowledge error (see section 2) of (P', V'). Now, we move from the hybrid to the simulated run where, in the PRS preamble, D did not commit to a randomness which could explain his message v_j (but rather an unrelated randomness r). Hence, D would not necessarily possess the witness of his statement.

Using the above hybrid argument, it can be shown that:

$$\Pr[\mathbb{P}^* \text{ can distinguish this simulation from a real run}] \leq$$

$$\Pr[\mathbb{P}^* \text{can break the ZK condition of } (P', V')] +$$

$$\Pr[\mathbb{P}^* \text{can break any of the commitments during the PRS preamble}] \leq$$

$$k^2 h_{com} + e'_z$$

$\mathbb{P}^*$ will see t of these simulations from D. Thus we can use the union bound and get that the probability that $\mathbb{P}^*$ will be able to distinguish any of the simulation from a real run is $(k^2 h_{com} + e'_z)t$.

Now, $\mathbb{V}$ will only accept in the protocol if the internal V he is running accepts $p_1, v_1, \ldots, p_t, v_t$. Recall that the probability that $\mathbb{V}$ accepts when interacting with $\mathbb{P}^*$ is ϕ. Thus the probability that V will accept an interaction with D who is running $\mathbb{P}^*$ can be computed as follows:

$$\Pr[V \text{accepts}] \geq$$

$$1 - \Pr[(\mathbb{P}^* \text{ does distinguish}) \vee (\mathbb{V} \text{ does not accept})] \geq$$
$$1 - \big(\Pr[\mathbb{P}^* \text{ does distinguish}] + \Pr[\mathbb{V} \text{ does not accept}]\big) \geq$$
$$1 - ((k^2 h_{com} + e'_z)t + (1 - \phi))$$

This value must be less than the soundness error of (P, V). Thus we get an upper bound on the soundness error of the compiled protocol

$$\phi \leq e_s + (k^2 h_{com} + e'_z)t$$

Note that if e_s, h_{com}, e'_z are all negligible and t, k are at most polynomial, the soundness error of the compiled protocol will be negligible.

Concurrent Statistical Zero-knowledge Lets consider an arbitrary unbounded concurrent verifier strategy. Let $\mathbb{V}^*$ be one of the verifiers representing a session in the concurrent verifier strategy. Given S, the simulator for the underlying protocol (P, V), we show how to construct a simulator $\mathbb{S}$ for the protocol $(\mathbb{P}, \mathbb{V})$. $\mathbb{S}$ will output a simulated transcript from a distribution which is only a negligible statistical distance from the distribution of the transcript of a real interaction. The simulator $\mathbb{S}$ is described formally in Figure 3.

$\mathbb{S}$ will first run S, the simulator of the underlying protocol. $\mathbb{S}$ will act as the honest verifier oracle for S recording all the randomness that he uses as the oracle. After running S, $\mathbb{S}$ will have a transcript $\hat{p}_1, \hat{v}_1, \ldots \hat{p}_t, \hat{v}_t$ and the randomness $\hat{r}$ (used in creating the honest verifier responses $\hat{v}_1, \ldots \hat{v}_t$). This transcript $\hat{p}_1, \hat{v}_1, \ldots \hat{p}_t, \hat{v}_t$ will be statistically close to a real run of (P, V).

As shown in the figure, $\mathbb{S}$ then runs the concurrently extractable simulator CEC-Sim (or in other words, the PRS simulator) and recovers the committed randomness r^* with probability at least $(1 - \epsilon)$. Since the commitments that $\mathbb{V}^*$ used during the PRS preamble are statistically binding, even an all powerful $\mathbb{V}^*$ will not be able to change them except with negligible probability. We call this probability b_{com}. After finishing the preamble, $\mathbb{S}$ will be a straightline simulator and will not rewind $\mathbb{V}^*$ any further.

$\mathbb{S}$ will now give $\mathbb{V}^*$ a string r' such that $r^* \oplus r' = \hat{r}$. Note that the distribution of r' will look completely uniform to $\mathbb{V}^*$ since $\mathbb{V}^*$ has no information about $\hat{r}$.

Now for each round of the protocol, the simulator will proceed as follows. In round j, $\mathbb{S}$ will give $\hat{p}_j$ to $\mathbb{V}^*$. Since $\mathbb{V}^*$ has already committed to r^*, it will now be forced use randomness $r^* \oplus r'$ which is exactly $\hat{r}$, . It will therefore be

Input: $\mathbb{V}^*$, one of the verifiers in an arbitrary unbounded concurrent verifier strategy.

The simulator $\mathbb{S}$

1. $\mathbb{S}$ acts as an honest verifier V and runs the simulator S of the argument system (P, V) on itself. $\mathbb{S}$ generates $\hat{r} \xleftarrow{r} R$ and uses it as randomness to interact with S. After the interaction, $\mathbb{S}$ gets as output the simulated transcript $\hat{p}_1, \hat{v}_1, \ldots \hat{p}_t \hat{v}_t$.
2. $\mathbb{S}$ runs a copy of $\mathbb{V}^*$
3. $\mathbb{S}$ runs the concurrently extractable simulator CEC-Sim on $\mathbb{V}^*$. CEC-Sim executes the PRS preamble with $\mathbb{V}^*$ and extracts its committed randomness r^*.
4. $\mathbb{S}$ carries out (P', V') with $\mathbb{V}^*$ in which $\mathbb{V}^*$ proves that all the shares it committed to in the PRS preamble are consistent with r^*.
5. $\mathbb{S}$ computes r' such that $r^* \oplus r' = \hat{r}$ and sends it to $\mathbb{V}^*$.
6. For $j = 1, \ldots t$:
   (a) $\mathbb{S}$ sends $\hat{p}_j$ to $\mathbb{V}^*$ and receives $\mathbb{V}^*$'s response $\hat{v}'_j$.
   (b) $\mathbb{S}$ carries out (P', V') with $\mathbb{V}^*$ in which $\mathbb{V}^*$ proves that its response $\hat{v}'_j = V(T^V_j, \hat{r})$. $\mathbb{S}$ aborts if $\hat{v}'_j \neq \hat{v}_j$.

Fig. 3. The simulator $\mathbb{S}$ for $(\mathbb{P}, \mathbb{V})$.

forced to respond with $\hat{v}_j$, except of course with the probability that he can break either the binding property of the commitment or the soundness of the zero-knowledge proof (P', V'). Since we are using statistically binding commitments and a zero knowledge *proof*, the probability of an all powerful adversary breaking the binding property of the commitments or the soundness property of the (P', V') is negligible. Thus the randomness that $\mathbb{V}^*$ is forced to use will be $\hat{r}$ and his response will therefore be $\hat{v}_j$, exactly as in the transcript created by S. If this is not the case, $\mathbb{S}$ aborts.

We now analyze the probability of failure of the simulator $\mathbb{S}$. From a union bound, we can directly bound this probability by analyzing the probability of all the events which may cause $\mathbb{S}$ to fail. The failure probability is upper bounded by:

$$\Pr[\text{Output of } S \text{ is not identically distributed to } (P, V)] +$$

$$\Pr[\text{CEC-Sim is unsuccessful in recovering } r^*] +$$

$$\Pr[\mathbb{V}^* \text{ breaks the binding property of any of the commitments}] +$$

$$\Pr[\mathbb{V}^* \text{ breaks the soundness property of } (P', V') \text{ for any of the executions}]$$

$$= \epsilon + e_z + k^2 b_{\text{com}} + e'_s t$$

Thus $\Delta((\mathbb{P}, \mathbb{V}^*)(x), \mathbb{S}^{\mathbb{V}^*(x)}) = (\epsilon + e_z + k^2 b_{\text{com}} + e'_s t)$ as claimed.

Note that if $\epsilon, e_z, b_{\text{com}}, e'_s$ are all negligible and t, k are at most polynomial, the simulated transcript will have negligible statistical difference from a real run of the protocol. ■

4.2 Concurrent statistical zero-knowledge arguments from any one way function

In order to build concurrent statistical zero-knowledge arguments from a OWF, we need the following theorem implicit in [NOV06].

Theorem 2 *If one way functions exist, every language in* **NP** *has a public-coin statistical zero-knowledge argument system.*

We can now apply our compiler to the protocol of Nguyen et al [NOV06] to get the following corollary.

Corollary 1 *If one way functions exist, every language in* **NP** *has a concurrent statistical zero-knowledge argument system.*

References

[BCC88] Gilles Brassard, David Chaum, and Claude Crépeau. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci.*, 37(2):156–189, 1988.

[BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-np have short interactive proofs? *Inf. Process. Lett.*, 25(2):127–132, 1987.

[BKK90] Joan Boyar, S. A. Kurtz, and Mark W. Krentel. A discrete logarithm implementation of perfect zero-knowledge blobs. *J. Cryptology*, 2(2):63–76, 1990.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *STOC*, pages 409–418, 1998.

[Fei90] Uriel Feige. Ph.d. thesis, alternative models for zero knowledge interactive proofs. Weizmann Institute of Science, 1990.

[GK96] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for np. *J. Cryptology*, 9(3):167–190, 1996.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in np have zero-knowledge proof systems. *J. ACM*, 38(3):691–729, 1991.

[Gol01] Oded Goldreich. *Foundations of Cryptography - Basic Tools.* Cambridge University Press, 2001.

[HHK$^+$05] Iftach Haitner, Omer Horvitz, Jonathan Katz, Chiu-Yuen Koo, Ruggero Morselli, and Ronen Shaltiel. Reducing complexity assumptions for statistically-hiding commitment. In *EUROCRYPT*, pages 58–77, 2005.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HR07] Iftach Haitner and Omer Reingold. Statistically-hiding commitment from any one-way function. In David S. Johnson and Uriel Feige, editors, *STOC*, pages 1–10. ACM, 2007.

[KP01] Joe Kilian and Erez Petrank. Concurrent and resettable zero-knowledge in poly-loalgorithm rounds. In *STOC*, pages 560–569, 2001.

[MOSV06] Daniele Micciancio, Shien Jin Ong, Amit Sahai, and Salil P. Vadhan. Concurrent zero knowledge without complexity assumptions. In *TCC*, pages 1–20, 2006.

[MP03] Daniele Micciancio and Erez Petrank. Simulatable commitments and efficient concurrent zero-knowledge. In *EUROCRYPT*, pages 140–159, 2003.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *J. Cryptology*, 4(2):151–158, 1991.

[NOV06] Minh-Huyen Nguyen, Shien Jin Ong, and Salil P. Vadhan. Statistical zero-knowledge arguments for np from any one-way function. In *FOCS*, pages 3–14. IEEE Computer Society, 2006.

[NOVY98] Moni Naor, Rafail Ostrovsky, Ramarathnam Venkatesan, and Moti Yung. Perfect zero-knowledge arguments for p using any one-way permutation. *J. Cryptology*, 11(2):87–108, 1998.

[NV06] Minh-Huyen Nguyen and Salil P. Vadhan. Zero knowledge with efficient provers. In *Proceedings of the 38th Annual ACM symposium on Theory of Computing*, 2006.

[NY89] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *STOC*, pages 33–43, 1989.

[Ost91] Rafail Ostrovsky. One-way functions, hard on average problems, and statistical zero-knowledge proofs. In *Structure in Complexity Theory Conference*, pages 133–138, 1991.

[OW93] Rafail Ostrovsky and Avi Wigderson. One-way fuctions are essential for non-trivial zero-knowledge. In *ISTCS*, pages 3–17, 1993.

[PRS02] Manoj Prabhakaran, Alon Rosen, and Amit Sahai. Concurrent zero knowledge with logarithmic round-complexity. In *FOCS*, pages 366–375, 2002.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *EUROCRYPT*, pages 415–431, 1999.