# Concurrent Zero Knowledge: Simplifications and Generalizations

Rafael Pass* Wei-Lung Dustin Tseng† Muthuramakrishnan Venkitasubramaniam‡

May 7, 2008

**Abstract**

Few techniques for obtaining concurrent zero-knowledge exist; all require a complex and subtle analysis. We provide an arguably simpler and more general analysis of the oblivious simulation technique of Kilian and Petrank (STOC'01) while achieving the same bounds as Prabhakaran, Rosen and Sahai (FOCS'02). Using this analysis, and relying on tools recently developed by Ong and Vadhan (Eurocrypt'07, TCC'08) we are able to establish the following *unconditional* results:

- every language in $\mathcal{NP}$ which has a $\mathcal{ZK}$ proof (resp. $\mathcal{ZK}$ argument, statistical $\mathcal{ZK}$ argument) also has a black-box concurrent $\mathcal{ZK}$ proof (resp. $\mathcal{ZK}$ argument, statistical $\mathcal{ZK}$ argument).
- every languge which has a statistical $\mathcal{ZK}$ proof also has an $\omega(\log n)$-round black-box concurrent statistical $\mathcal{ZK}$ proof.

**Keywords:** zero-knowledge, concurrency, oblivious simulation.

*Cornell University, E-Mail: rafael@cs.cornell.edu
†Cornell University, E-Mail: wdtseng@cs.cornell.edu
‡Cornell University, E-Mail: vmuthu@cs.cornell.edu

# 1 Introduction

Following the seminal works of [DDN, FS] from the early 90's, concurrent security of cryptographic protocols has been an active area of research. Yet, it is still not well-understood under what circumstances concurrent security can be achieved. One potential reason for this might be the complexity of traditional analyses. In this work we focus on simplifying (and generalizing) the analysis of concurrent security in one of the most basic settings, namely that of *zero-knowledge proofs*.

Zero-knowledge ($\mathcal{ZK}$) interactive proofs [GMR] are paradoxical constructs that allow one player (called the Prover) to convince another player (called the Verifier) of the validity of a mathematical statement $x \in L$, while providing *zero additional knowledge* to the Verifier. Beyond being fascinating in their own right, $\mathcal{ZK}$ proof have numerous cryptographic applications and are one of the most fundamental cryptographic building blocks. As such, techniques developed in the context of $\mathcal{ZK}$ often extend to more general types of interactions.

The notion of concurrent $\mathcal{ZK}$-knowledge, first introduced and achieved in the paper by Dwork, Naor and Sahai [DNS], considers the execution of zero-knowledge proofs in an asynchronous and concurrent setting. More precisely, we consider a single adversary mounting a coordinated attack by acting as a verifier in many concurrent executions (called sessions). Since the original protocols by Dwork, Naor and Sahai (which relied on so called "timing assumptions"), various other concurrent $\mathcal{ZK}$ have been obtained based on different set-up assumptions (e.g., [DS] [D00] [CGGM]). On the other hand, in the standard model without set-up assumptions, Canetti, Kilian, Petrank and Rosen [CKPR] (building on earlier works by [KPR] [R00]) show that concurrent $\mathcal{ZK}$ proofs for non-trivial languages, with so called "black-box" simulators, require at least $\Omega(\frac{\log n}{\log \log n})$ number of communication rounds. Richardson and Kilian [RK] constructed the first concurrent $\mathcal{ZK}$ argument in the standard model without any extra set-up assumptions. Their protocol, which uses a black-box simulator, requires $O(n^\epsilon)$ number of rounds. Kilian and Petrank [KP] then introduced a new *oblivious* simulator. Using this technique they obtained a simpler and cleaner analysis, and additionally improved the round complexity to $\tilde{O}(\log^2 n)$. Finally, the work of Prabhakaran, Rosen and Sahai [PRS] further simplifies and improves the analysis of the oblivious simulator, obtaining an essentially optimal round complexity of $\tilde{O}(\log n)$.

However, despite these simplifications and improvements, the analysis remains complex, making it hard to employ more broadly.[1]

*Simplifications and generalizations.* In this paper we provide a new analysis of the KP oblivious simulator. Although on the outset our analysis follows the same high-level structure as that of [PRS], our techniques simplify the proof (while at the same time providing a more general analysis). Our first result is a new proof of the following theorems originally shown in [PRS]:

**Theorem 1.** *Assuming the existence of one-way functions, every language in $\mathcal{NP}$ has a $\omega(\log n)$ round black-box concurrent $\mathcal{ZK}$ argument.*

**Theorem 2.** *Assuming the existence of 2-round statistically hiding commitments, every language in $\mathcal{NP}$ has a $\omega(\log n)$ round black-box concurrent $\mathcal{ZK}$ proof.*

[1] For instance, several subsequent results rely on the "inner-workings" of the PRS analysis and in particular rely on claims implicit in [PRS] (see e.g., [MOSV, BPS, GMOS, HKKL].)

Our analysis also permits us to easier analyze more protocols.[2] In particular, whereas previous analyses typically required the use *non-interactive* commitments in the zero-knowledge protocol, our analysis handles zero-knowldge protocols which use also multi-round commitments. First, our protocol for $\mathcal{ZK}$ arguments directly yields the following theorem recently shown by [GMOS]:[3]

**Theorem 3.** *Assuming the existence of one-way functions and the existence of a $t(n)$-round statistical $\mathcal{ZK}$ argument for $\mathcal{NP}$, every language in $\mathcal{NP}$ has a $t(n) + \omega(\log n)$ round statistical black-box concurrent $\mathcal{ZK}$ argument.*

Combined with the recent result of [NOV], the above theorem shows that statistical concurrent $\mathcal{ZK}$ arguments can be based only on one-way functions.

*New results.* Turning to new results, our analysis for $\mathcal{ZK}$ proofs also establishes the following generalization of Theorem 2:

**Theorem 4.** *Assuming $t(n)$-round statistically hiding commitments, every language in $\mathcal{NP}$ has a $O(t(n)) + \omega(\log n)$ round black-box concurrent $\mathcal{ZK}$ proof.*

Combined with the recent results by [NOV, HR] this establishes that concurrent $\mathcal{ZK}$ knowledge *proofs* (as opposed to arguments) can be based on only one-way functions. Additionally, following a suggestion by Ong and Vadhan [OV08] and relying on their "instance-based" commitments, we can complete the characterization (initiated in [MOSV]) of languages having concurrent *statistical $\mathcal{ZK}$ proofs*.[4]

**Theorem 5.** *Every language that has a statistical $\mathcal{ZK}$ proof also has a $\omega(\log n)$-round black-box statistical concurrent $\mathcal{ZK}$ proof.*

Finally, by extending the results of [OV07, OV08], we also obtain unconditional characterizations of computational $\mathcal{ZK}$ and statistical $\mathcal{ZK}$ arguments.

**Theorem 6.** *Every language in $\mathcal{NP}$ that has a computational $\mathcal{ZK}$ proof (resp. statistical $\mathcal{ZK}$ argument, computational $\mathcal{ZK}$ argument) also has a black-box concurrent computational $\mathcal{ZK}$ proof (resp. statistical $\mathcal{ZK}$ argument, computational $\mathcal{ZK}$ argument).*

For the case of statistical $\mathcal{ZK}$ arguments our protocol has essentially optimal round-complexity, namely $\omega(\log n)$ rounds. For the other results we obtain $O(t(n))\omega(\log n)$ rounds, where $t(n)$ is the round-complexity of any construction of statistically-hiding commitments from one-way functions [NOV, HR].

*Our Techniques.* Kilian and Petrank's (KP) ingenious concurrent simulation technique relies on a static—and oblivious—rewinding schedule; namely, the simulator rewinds the verifier after some fixed number of messages, independent of the content and the scheduling of the messages (and in particular what session these messages belong too). The crux of their proof is then to show

[2] The analysis of [PRS] considers only so called "committed-verifier" protocols which are implemented using non-interactive commitments. See [MOSV] for more detail.

[3] In contrast, [GMOS] were required to consider a quite different protocol to establish this result.

[4] In [OV07], Ong and Vadhan suggested that their instance-based commitments could prove useful in establishing this results. However, as they point out in [OV07], their work did not directly extend to yield this result because the analysis of [MOSV] and [PRS] only considered non-interactive commitments, whereas the instance-based commitments constructed in [OV07] require multiple rounds.

that in this schedule, every session is "successfully rewound" at least once with high probability; in a successful rewind, the simulator can extract a "trapdoor" that will allow it to complete the simulation. To bound the failure probability, they rely on a subtle computation of conditional probabilities. Prabhakaran, Rosen and Sahai (PRS) [PRS, R04, PS], on the other hand, directly analyze the probability space, i.e. the random tapes of the simulator; this makes the analysis both simpler and sharper. The elegant high level idea is to show that each "bad" random tape (that produces a failed simulation) can be mapped into super-polynomially many *distinct* "good" tapes. This is done by identifying random tape segments, called *rewinding intervals*, that can be "swapped" in order to turn a bad tape into a good one[5]. The crux of their proof is then to count how many such "swappings" actually generate new and distinct random tapes. However, the analysis is complicated by the fact that swappings performed on different rewinding intervals may overlap and even remove other possible rewinding intervals.[6] In the end, the PRS proof relies on a complex *global* analysis to first count the number of swappings available to each rewinding interval, and subsequently lower-bound the overall number of available swappings; see Appendix A for a slightly more detailed overview of the PRS analysis.

Our main contribution is a new way of mapping good tapes to bad tapes (and just like in [KP, PRS] thus bounding the failure probability of the simulator). Towards this goal, we identify a *stronger* notion of rewinding intervals called *composable blocks*. Just like rewinding intervals, properties of composable blocks guarantee that a "swap" will generate a new good random tape; moreover, these same properties are closed under *composition*—namely the swapping of one such block leaves other composable blocks intact. By this new composition property, it is enough to identify $K$ composable blocks to conclude that the simulation fails with probability less than $2^{-K}$. In essence, our proof will consist of two simple steps: First, we establish "local" properties of a composable block (namely that a swap generates one new good random tape, and that swappings are composable); then, we count the number of composable blocks on a bad random tape. As we shall see, each round in the protocol gives rise to a new composable block; as such, our analysis conveys a strong intuition of how "each additional round of the protocol halves the simulator's failing probability".

To employ this new notion of composable blocks, we additionaly consider and analyze a "lazy" variant of the KP simulator. Intuitively, the "lazy KP simulator" is identical to the KP simulator but only makes use of information gathered in its rewindings after some delay. The lazy KP simulator can only fail more often than the original KP simulator (and thus our analysis indirectly also applies to the KP simulator); yet, considering this "weaker" simulator enables our way of analyzing the failure probability of the simulation. In a sense, much like making a stronger inductive hypothesis can enable the inductive step, our stronger notion of composable blocks and our weaker simulator enable the analysis.

On a technical level, our approach also differs from the one in [PRS] in that we directly analyze the failure probability of the original simulator, whereas PRS instead analyze a "hybrid" simulator. This feature facilitates analysis for a more general class of protocols (which will be required to reach our results).

*Overview.* After some definitions in Section 2, we introduce our protocol for concurrent $\mathcal{ZK}$ argu-

[5]Here we use the terminologies from Rosen's thesis [R04].

[6]We mention that the proceedings version of [PRS] gives only a sample illustration of multiple swappings on a rewinding interval. See page 84 and page 90 in [R04] for examples of how overlapping invervals complicate the analysis.

ments in Section 3, and prove Theorem 1 and 3 in Section 3.2. We follow up with the construction of concurrent $\mathcal{ZK}$ proofs in Section 4. Finally, we give our unconditional characterizations of concurrent zero-knowledge. We provide a brief overview of the PRS analysis in Appendix A. Appendix B contains formal definitions and results related to instance-based commitments.

# 2 Definitions

We assume familiarity with indistinguishability and interactive proofs.

## 2.1 Black-box Concurrent Zero-Knowledge

Let $\langle P, V \rangle$ be an interactive proof for a language $L$. Consider a concurrent adversarial verifier $V^*$ that on common input $x$ and auxiliary input $z$, interacts with $m(|x|)$ independent copies of $P$ concurrently, without any restrictions over the scheduling of the messages in the different interactions with $P$. Let $View^P_{V^*}(x, z)$ denote the random variable describing the view of the adversary $V^*$ in an interaction with $P$.

**Definition 1.** *Let $\langle P, V \rangle$ be an interactive proof system for a language $L$. We say that $\langle P, V \rangle$ is* black-box concurrent zero-knowledge *if for every polynomials $q$ and $m$, there exists a probabilistic polynomial time algorithm $S_{q,m}$, such that for every concurrent adversary $V^*$ that on common input $x$ and auxiliary input $z$ opens up $m(|x|)$ executions and has a running-time bounded by $q(|x|)$, $S_{q,m}(x, z)$ runs in time polynomial in $|x|$. Furthermore, it holds that the ensembles $\{View^P_{V^*}(x, z)\}_{x \in L, z \in \{0,1\}^*}$ and $\{S_{q,m}(x, z)\}_{x \in L, z \in \{0,1\}^*}$ are computationally indistinguishable over $x \in L$.*

## 2.2 Other Primitives

We informally define the other primitives we use in the construction of our protocol.

**Witness-Indistinguishable Proofs [FS]:** An interactive proof is said to be *witness indistinguishable* ($\mathcal{WI}$) if the verifier's view is "computationally independent" of the witness used by the prover for proving the statement—i.e. the views of the verifier in interactions with provers using two different witnesses are indistinguishable. Zero-knowledge proofs are automatically $\mathcal{WI}$.

**Statistical Special-Sound Proofs [CDS]:** A $k$-round interactive proof for language $L \in \mathcal{NP}$ with witness relation $R_L$ and security parameter $n$ is statistical special-sound (SS) with respect to $R_L$ if the following holds: There exists a deterministic polynomial time procedure that can extract a witness with overwhelming probability in $n$ given an randomly sampled $(k-2)$-message prefix of the protocol ($\vec{\alpha}$) (of an interaction between an (unbounded) prover and an honest verifier) and two independent accepting completions of this prefix ($(\vec{\alpha}, \beta, \gamma)$ and $(\vec{\alpha}, \beta', \gamma')$). 3-round SS-$\mathcal{WI}$ proofs for $\mathcal{NP}$ can be based on one-way permutation (by relying on Blum's HC protocol [Blum] implemented using non-interactive commitment) and 4-round SS-$\mathcal{WI}$ proof can be based on only one-way functions (by instead using two-rounds commitments [Naor]).

**Proofs of Knowledge [FS, BG]:** An interactive proof is a proof of knowledge if the prover convinces the verifier not only of the validity of a statement, but also that it possesses a witness for the statement.

# 3 Concurrent Zero-Knowledge Arguments

## 3.1 Protocol for Concurrent Zero-Knowledge Arguments

Our concurrent $\mathcal{ZK}$ protocol ConcZKArg (also used in [PV]) is a slight variant of the precise $\mathcal{ZK}$ protocol of [MP], which in turn is a modification of the Feige-Shamir protocol [FS]. The protocol proceeds in the following three stages, given the common input statement $x \in \{0,1\}^*$, the security parameter $n$, and a "round-parameter" $k$:

1. In Stage Init, the Verifier picks two random strings $s_1, s_2 \in \{0,1\}^n$, and sends their images $c_1 = f(s_1)$, $c_2 = f(s_2)$ through a one-way function $f$ to the Prover. The Prover and the Verifier also exchange all but the last two messages $\vec{\alpha}_1, \ldots, \vec{\alpha}_k$ of $k$ invocations of a $\mathcal{WI}$ statistical special-sound proof of knowledge of the fact that either $c_1$ or $c_2$ is in the image set of $f$. The end of Stage Init is called the START of the protocol.

2. Stage 1 consists of $k$ iterations of message exchange where in the $j^{th}$ iteration, the Prover sends $\beta_j \leftarrow \{0,1\}^*$, a random second last message for the $j^{th}$ proof of knowledge, and the Verifier replies with the last message $\gamma_j$ of the proof. These $k$ iterations are called *slots*. A slot is *convincing* if the Verifier succeeds producing an accepting proof. If there is ever an *unconvincing* slot, the Prover aborts the whole session. The end of Stage 1 (after $k$ convincing slots) is called the END of the protocol.

3. In Stage 2, the Prover provides a (statistical) $\mathcal{WI}$ proof of knowledge of the fact that either $x$ is in the language, or at least one of $c_1$ and $c_2$ are in the image set of $f$.

The completeness and the soundness of the protocol follows directly from the proof of Feige and Shamir [FS]; in fact, the protocol is an instantiation of theirs. Intuitively, to cheat in the protocol a prover must "know" an inverse to either $c_1$ or $c_2$ (since Stage 2 is an argument of knowledge), which requires inverting the one-way function $f$.

## 3.2 The $\mathcal{ZK}$ simulator

We show that whenever $k$ is super logarithmic (i.e. $k = \omega(\log n)$), our protocol is concurrent $\mathcal{ZK}$. On a very high-level the simulation follows that of Feige and Shamir [FS]: the simulator will attempt to rewind one of the *slots* (i.e. the special-sound proofs). If the simulator receives two accepting proof transcripts, the special-soundness property allows the simulator to extract a "fake" witness $s_i$ such that $c_i = f(s_i)$. This witness can later be used in the second phase of the protocol.

Towards this goal, we provide an oblivious black-box simulator similar to [KP]. We describe a procedure $\mathsf{Sim}(x,z) = \mathsf{Sim}^{V^*}(x,z)$ that given input instance $x$, auxiliary input $z$, and black-box access to $V^*$, outputs a view that is indistinguishable from the real view of $V^*(x,z)$.

*Description of* Sim. Let $n$ be the security parameter, $m$ be a bound on the number of concurrent sessions invoked by $V^*$ and $T$ be the total number of messages exchanged, bounded by $O(mk)$, a

PROTOCOL CONCZKARG

**Common Input:** an instance $x$ of a language $L$ with witness relation $R_L$.

**Auxiliary Input for Prover:** a witness $w$, such that $(x, w) \in R_L(x)$.

**Stage Init:**

V uniformly chooses $s_1, s_2 \in \{0,1\}^n$.

V $\rightarrow$ P: $c_1 = f(s_1), c_2 = f(s_2)$.

V $\leftrightarrow$ P: Exchange (interactively) all but the last two messages $\vec{\alpha}_1, \ldots, \vec{\alpha}_k$ of $k$ statistical $\mathcal{WI}$ special-sound proofs of knowledge (SS-POK) of the statement:

*either* there exists a value $s$ s.t. $c_1 = f(s)$

*or* there exists a value $s$ s.t. $c_2 = f(s)$

The proof of knowledge is with respect to the witness relation $R'_L = \{((x_1, x_2), (y_1, y_2)) : f(x_1) = y_1 \text{ or } f(x_2) = y_2\}$

We say the protocol has reached START (of Stage 1) if all messages in Stage Init are exchanged.

**Stage 1:**

For $j = 1$ to $k$ do

P $\rightarrow$ V: The second last message $\beta_j$ of the $j^{\text{th}}$ $\mathcal{WI}$ SS-POK.

V $\rightarrow$ P: The last message $\gamma_j$ of the $j^{\text{th}}$ $\mathcal{WI}$ SS-POK.

We say the protocol has reached END (of Stage 1) if all $k$ SS-POKs are accepted.

**Stage 2:**

P $\leftrightarrow$ V: a (statistical) $\mathcal{WI}$ argument of knowledge from $P$ to $V$ of the statement

*either* there exists values $s'_1, s'_2$ s.t. either $c_1 = f(s'_1)$ or $c_2 = f(s'_2)$.

*or* $x \in L$

The argument of knowledge is with respect to the witness relation $R_{L \vee L'}(c_1, c_2, x) = \{(s'_1, s'_2, w) : (s'_1, s'_2) \in R_{L'}(c_1, c_2) \vee w \in R_L(x)\}$.

Figure 1: Concurrent $\mathcal{ZK}$ argument for $\mathcal{NP}$ with parameter $k$

polynomial. In order to extract the "fake witness" ($s_1$ or $s_2$) from $V^*$, Sim will follow an oblivious rewinding schedule based on the number of messages exchanged so far, just like in [KP] and [PRS].

More precisely, Sim will use the helper-procedure recursive-rewind, formally described in Figure 2. recursive-rewind$(t, s, h)$ simulates $V^*(x, z)$ for $t$ messages starting from state $s$ of $V^*$ as follows:

1. From state $s$ recursively simulate $V^*$ for $t/2$ messages; let $s_1$ be the resulting state of $V^*$.
2. Rewind $V^*$ back to state $s$ and recursively simulate $V^*$ for $t/2$ messages.
3. From $s_1$ (the state at the end of step 1), recursively simulate $V^*$ for $t/2$ messages.
4. Rewind $V^*$ back to state $s_1$ and recursively simulate $V^*$ again for $t/2$ messages.

At the base case of the recursion ($t = 1$), recursive-rewind acts like an honest prover during Stage 1 of the protocol. Meanwhile, recursive-rewind remembers the messages sent by $V^*$ during the entire simulation in the "history" variable $h$. When the simulation reaches END, recursive-rewind will attempt to extract the "fake witness" ($s_1$ or $s_2$) from $V^*$ using messages stored in $h$, and use it for the $\mathcal{WI}$-POK in stage 2. If the extraction is unsuccessful, recursive-rewind will output $\perp$, and we say Sim is stuck (Sim will output $\perp$ and stop the simulation as well). Otherwise, Sim will output the view of $V^*$ on the output thread of recursive-rewind$(T, s_0, \emptyset)$, where $s_0$ is the initial state of $V^*$.

We now introduce a few terminologies regarding the recursive structure of recursive-rewind. We refer to each call of recursive-rewind as a *block*, and call a sequence of blocks that simulate consecutive parts of the protocol a *thread* (motivated by Figure 3). Because recursive-rewind makes four recursive calls, each block would contain four smaller blocks of half the size. Of these four blocks, we call the first pair (respectively the last pair) *siblings*, since these two blocks share the same "parent" start state. In addition, sibling blocks also share the same "history" of recorded messages. That is, during the simulation of a block, we do *not* use the information obtained during the simulation of its sibling block, so that the two blocks are totally symmetric. Since our simulator does not update $h$ until the simulation in both siblings are finished, we call it "lazy".

In addition to this lazy modification, we have changed how blocks are threaded together from [KP] and [PRS]. In recursive-rewind, the recursive calls for the second half of messages are continued from the first recursive execution of the first half of messages. That is, the second set of recursive calls are continued from state $s_1$, similar to the precise simulation of [MP] and [PPSTV]. [KP] and [PRS], in contrast, continues the recursive calls from state $s_2$. Due to the new symmetry introduced by lazy simulation, either choice will work with our analysis. In the end, recursive-rewind outputs the thread connecting the first recursive call to each half of the messages; this thread is called the *output thread*. See Figure 3 for an illustration of blocks and threads in an execution of recursive-rewind.

### 3.3 Proof Overview

In order to prove the correctness of the simulation, we need to show that for every adversarial verifier $V^*$, the simulator runs in polynomial time and the output distribution is "correct". To analyze the running time of the simulator, it suffices to compute the number of messages exchanged, since we can assume that a maximum of $poly(n)$ time is spent on processing each message. It follows from the recursive structure of the simulator that the number of messages exchanged is doubled for each level of the recursion; since we have a recursive depth of $\log_2 T$, the running time of the simulator is $poly(n) \cdot T \cdot 2^{\log_2 T} = poly(n) \cdot T^2 = poly(n)$.

recursive-rewind$(t, s, h)$:

1. **Base Case:** $t = 1$
   (a) If the next scheduled message is from an aborted session (i.e. there has been an unconvincing slot) return $(s, h)$ (i.e. do nothing).
   (b) If the next scheduled message is a Stage 1 prover message for session $i$, pick a random message $\beta \leftarrow \{0,1\}^{n^2}$ just like an honest prover. Send $\beta$ to $V^*$. Suppose $V^*$ produces the response $v$ intended for session $j$. Update state $s$ to be the new state of $V^*$, and update the history $h$ with the messages $\beta$ and $v$. If $v$ is an unconvincing last message of a proof of knowledge a SS-POK in Stage 1, abort session $j$.
   (c) If we reach the END of a session, attempt to extract (and remember) a fake witness of the session using previous messages stored in $h$. If the extraction is unsuccessful, output $\perp$.
   (d) If the next scheduled message is a Stage 2 prover message for session $i$, use the extracted fake witness to complete the *witness-indistinguishable proof of knowledge*.
2. **Recursive step**
   # Rewind first half twice
   (a) $(s_1, h_1) \leftarrow$ recursive-rewind$(t/2, s, h)$
   (b) $(s_2, h_2) \leftarrow$ recursive-rewind$(t/2, s, h)$
   # Rewind second half twice from state $s_1$
   (c) $(s_3, h_3) \leftarrow$ recursive-rewind$(t/2, s_1, h_1 \cup h_2)$
   (d) $(s_4, h_4) \leftarrow$ recursive-rewind$(t/2, s_1, h_1 \cup h_2)$
   (e) output $(s_3, h_3 \cup h_4)$ /* $s \rightarrow s_1 \rightarrow s_3$ is called the output thread */

Figure 2: The recursive procedure used by Sim—the "lazy" KP simulator.

The correctness of the output view follows from the fact that Stage 1 messages are chosen uniformly at random, and that the protocol used in Stage 2 is witness indistinguishable. Therefore, as long as Sim gets stuck with negligible probability, taken over the random tapes of Sim (the random tape of $V^*$ is fixed during black box simulation), the output distribution is correct. Towards this goal we will show that the probability of getting stuck at any point in the simulation is negligible.

Recall that Sim can only get stuck on a particular thread when the simulation reaches the END of some session for which the "fake" witness has not yet been extracted. We will show that the probability of getting stuck on *any session* and *any thread* is negligible. Since there are only polynomially many sessions and threads, the main theorem follows by the union bound.

Fix any thread $h$ and session $i$; we call them the "main" thread and the "main" session, and call all other threads and sessions "auxiliary". We say a random tape of Sim is bad if Sim gets stuck on thread $h$ and session $i$ (if at all); all other random tapes are called good (including those that got stuck on an auxiliary session or thread). The high-level idea, just like in [PRS], is to show that for every bad random tape, there exists super-polynomially many good random tapes. Furthermore, the good tapes corresponding to any two bad tapes are disjoint. Hence the probability of a tape being bad is negligible. From here on, START and END refer to those on the main session and thread unless otherwise noted.

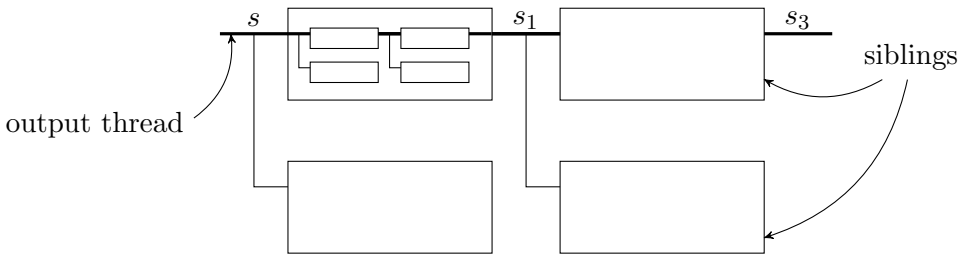



Figure 3: A pictorial representation of the rewind schedule. The boxes represent recursive blocks, and the lines represent threads of a executions. The output thread is highlighted, and is the thread finally output by Sim.

Given a bad random tape we show how to generate super-polynomially many good random tapes. Recall that on a bad tape, the simulator reaches END without extracting a "fake witness". Hence, it has convincing runs of all slots in the main thread (to reach END), but no convincing runs of any slot in an auxiliary thread prior to END (so that Sim cannot extract a witness). Intuitively, to generate a good tape from a bad one we just need to "swap" a convincing slot from the main thread into an auxiliary thread. After the swapping, should the simulation reaches END of the main session on the main thread, the convincing slot on the auxiliary thread, together with the corresponding convincing slot on the main thread, will allow Sim to extract "fake witness". Hence the simulation may continue on without getting stuck.

To actually "swap" convincing slots, we modify the random tape of Sim. The basic operation that we perform on the random tape is to exchange the randomness used by sibling blocks (i.e. exchange the portions of the random tape used to simulate these blocks). Since $V^*$ is deterministic and sibling blocks share the same start state, the messages produced by $V^*$ in sibling blocks are uniquely determined by the messages produced by Sim. But the actions of Sim are uniquely determined by its random tape and the history repository, and (due to the lazy property of Sim) siblings share the same history $h$. Thus, swapping the random tape between siblings swaps the simulation result in the two blocks *exactly*.

Intuitively, we call a block on the main thread composable if it satisfies the following properties:

***Goodness.*** Swapping a composable block with its sibling produces a good random tape.

***Composability.*** The above swap leaves other composable blocks on the main thread composable.

***Reversibility* (signature).** From the random tape obtained after swapping the blocks, we must be able to undo the swap operation and retrieve the original tape. More precisely, given the tape, we should be able to identify the block that has been swapped. The undo operation ensures that good tapes obtained from different bad tapes are different.

Consider $K$ composable blocks with an ordering such that each swap will leave the successive composable blocks still composable. Then, we can generate $2^K - 1$ good random tapes by choosing to swap each block or not in the ordering. By a simple counting argument, we will show that for any bad tape, there are $k - 2\log_2 T$ composable blocks with an ordering. Thus, from every bad tape we can generate $2^{k-2\log_2 T}$ distinct good tapes. The undo procedure also shows that different bad

tapes generate different good tapes. Thus, if $k \in \omega(\log_2 T) = \omega(\log n)$, the probability of having a bad tape is negligible.

## 3.4 The Actual Proof

Formally, Sim may output $\perp$ for two reasons. Firstly, it may reach END without encountering two convincing slots after the START of the session; we call this a *rewinding failure*. Secondly, Sim may not be able to extract a fake witness even though it has access to matching special-sound transcripts; we call this a *special-sound failure*. We first focus on bounding the probability of special-sound failures.

### 3.4.1 Composable Blocks

We first define the notion of composable blocks and show that they satisfy the three properties of *goodness, composability* and *reversibility*. Recall that a block is the scope of an execution of recursive-rewind, and therefore contains many branching threads. We say "the block contains the thread" or "the block is on the thread" interchangeably. Let us fix a particular main session and main thread, and formally define a random tape to be bad if Sim encounters a *rewinding failure* during the main session on the main thread; otherwise a random tape is good. From here on START and END refers to those of the main session and main thread, unless otherwise noted.

**Definition 2** (Composable Block)**.** *Consider an execution of* recursive-rewind *with a fixed random tape (not necessarily bad) and main thread $h$. A block $B$ with sibling $B'$ in the simulation is called a* composable block*, with respect to the main thread and session, if it satisfies the following conditions:*

Main block condition**:** *$B$ contains the main thread $h$, a convincing slot of the main session (not necessarily on the main thread) and does not contain* START*.*

Sibling condition**:** *$B'$ does not contain any* END *of the main session.*

Signature condition**:** *The simulation after* START *but before $B$ contains only convincing slots on the main thread $h$, and contains no convincing slots on the auxiliary threads.*

As we will soon see, the Main block condition and the Sibling condition implies goodness and composability, while the Signature condition implies reversibility.

We also define an ordering relation $>$ on composable blocks.

**Definition 3.** *Let $C$ and $B$ be two blocks on a common thread. We write $C > B$ iff*

- *$C$ and $B$ are disjoint, and $C$ occurs before $B$ (Case 1 in Figure 4), or*
- *$C$ and $B$ are not disjoint, and $C$ is a larger block that contains $B$ (Case 2 in Figure 4)*

Note that given two blocks on the same thread, if they are not disjoint, then one must contain another. Thus definition 3 defines a total order on any set of blocks that share a common thread.

Finally, we define an undo function on random tapes using the following deterministic procedure:

- Given a random tape $\tau'$, execute recursive-rewind with the tape $\tau'$. Call a block *special* if it contains a convincing slot of the main session on an auxiliary thread.

- Look for the first special block, $D$, after START; that is, any other special block $E$ after START satisfies $D > E$.
- Swap the parts of $\tau'$ used by $D$ and its sibling, and output the new random tape.

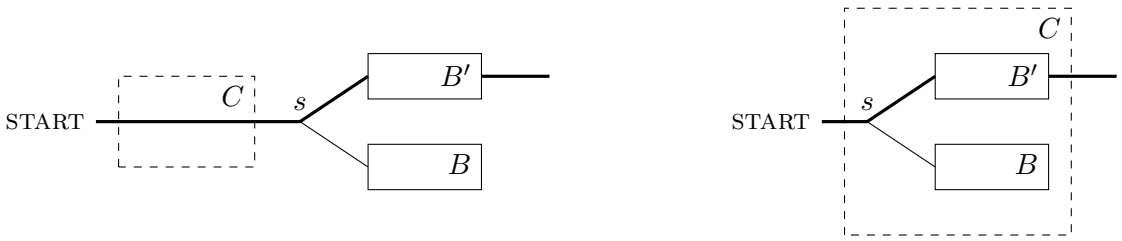
**Claim 1.** *Let $\tau$ be a random tape (not necessarily bad). Let $B$ be a* composable *block with sibling $B'$ when* recursive-rewind *is executed with random tape $\tau$, and let $s$ be the common start state of $B$ and $B'$. Further, let $\tau'$ be the random tape obtained after swapping the blocks $B$ and $B'$. Then:*

1. [Goodness]: *$\tau'$ is a* good *random tape.*
2. [Composability]: *Any* composable *block $C$ on $\tau$ with $C > B$ is still* composable *on $\tau'$.*
3. [Reversibility]: $\mathsf{undo}(\tau') = \tau$.

*Proof.* Recall that the simulation inside blocks $B$ and $B'$ are exchanged *exactly* after the swapping.

**Goodness** Let $s$ be the common start state of $B$ and $B'$. As shown in Figure 4, when recursive-rewind is executed with $\tau'$, $B'$ will now be on the main thread. Recall that $B'$ does not contain any END of the main session (sibling condition). Thus, if the END of the main session ever occurs on the main thread, it will occur after both $B$ and $B'$ are executed. In that case, both the convincing slot in $B$ and the corresponding convincing slot on the main thread occur before END and after the same START (see Figure 4). Thus, $\tau'$ is a good tape.

**Composability** Given a composable block $C > B$ with sibling $C'$ on $\tau$, we have two cases as shown in Figure 4. In case 1, when $C$ is disjoint from $B$, the swapping of $B$ and $B'$ does not change the simulation inside $C$, $C'$ (not shown), and between START and $C$. Respectively, this leaves the main block condition, sibling condition, and signature condition of $C$ intact on $\tau'$. On the other hand, in case 2 where $C$ contains $B$, the swapping of $B$ and $B'$ again leaves the simulation inside $C'$ (not shown) and between START and $C$ unchanged, keeping the sibling condition and signature condition intact. In addition, since $C$ still contains $B$ under $\tau'$, and $B$ in turn contains a convincing slot, the main block condition still holds as well (other parts of $C$ may have changed). In both cases, $C$ continues to be a composable block on $\tau'$.




Case 1: $C$ occurs before $B$ Case 2: $C$ contains $B$

Figure 4: The before-and-after block diagram of the swapping procedure in Claim 1. The main thread is shown in a thick line, and a typical composable block $C > B$ is shown in a dashed box.

**Reversibility** Finally, we need to show that $\mathsf{undo}(\tau') = \tau$. Clearly, after the swapping, block $B$ is no longer on the main thread and contains a convincing slot. On the other hand, any block $C$ earlier than $B$ and $B'$ either is simulated before $s$ or contains $B$ and $B'$ (as shown in figure 4). In Case 1, $C$ is not special on $\tau$ because of the signature condition of the composable block $B$, and thus is not special on $\tau'$ either. In Case 2, since $C$ contains $B'$, it is on the main thread and is not special. Thus, undo will always choose $D = B'$, and perform the inverse swapping on $\tau'$ to obtain $\tau$[7].

■

The next claim demonstrates how to compose multiple composable blocks.

**Claim 2.** *Let $\tau$ be a bad random tape, $\mathcal{B} = \{B_1, \ldots, B_p\}$ be a set of composable blocks for $\tau$. Then, we can generate a set of good random tapes, $S(\tau, \mathcal{B})$, by swapping the various composable blocks in $\mathcal{B}$, so that the following holds:*

1. $|S(\tau, \mathcal{B})| \geq 2^p - 1$.
2. *For any bad tape $\tau' \neq \tau$ and any set of composable blocks $\mathcal{B}'$ for $\tau'$, $S(\tau, \mathcal{B}) \cap S(\tau', \mathcal{B}') = \emptyset$.*

*Proof.* Since all composable blocks lie on a common thread (i.e. the main thread), there is a total ordering of the blocks. Without loss of generality, let $B_1 > B_2 \cdots > B_p$. Consider any non-empty subsequence of $1, \ldots, p$, say $u_1, \ldots, u_q$. There are $2^p - 1$ such sequences. Let $\tau_{u_1 \ldots u_q}$ be the random tapes obtained from $\tau$ by swapping the blocks $B_{u_i}$ with its sibling, in the order of $i = q, q-1, \ldots, 1$.

From Claim 1, it follows that $\tau_{u_1 \ldots u_q}$ is a good random tape. We further note that given $\tau_{u_1 \ldots u_q}$, we can recover the blocks $B_{u_1}, \ldots B_{u_q}$ by repeatedly applying undo until we reach a bad tape (it will always be $\tau$). Since undo is deterministic, we must have $\tau_{\vec{u}} \neq \tau_{\vec{v}}$ whenever $\vec{u} \neq \vec{v}$ in order for undo to recover a different set of blocks on input $\tau_{\vec{u}}$ and $\tau_{\vec{v}}$. Thus, we obtain $2^p - 1$ distinct good random tapes.

Similarly, take any $\alpha \in S(\tau, \mathcal{B})$ and $\beta \in S(\tau', \mathcal{B}')$. Applying undo repeatedly on $\alpha$ until the result is a bad tape will result in $\tau$, while applying the same procedure on $\beta$ will give $\tau'$. Again, since undo is deterministic, we have $\alpha \neq \beta$. ■

**Corollary 1.** *Suppose every bad random tape had $p$ composable blocks. Then, the probability of a random tape being bad is at most $\frac{1}{2^p}$.*

### 3.4.2 Number of Composable Blocks

We now proceed to count the number of composable blocks. First we introduce the notion of *minimal containing blocks*. For each slot, its *minimal containing block* is the minimal block on the main thread that contains the slot. Claims 3 and 4 below together show that there are at least $k - 2\log T$ composable blocks when we run Sim with a bad tape.

[7] Note that we here rely on the lazy property of Sim, and in particular the "exact" swapping of sibling blocks. Without this exact swapping, if the END of an auxiliary session occurs before the convincing slot in $B$, that END may cause Sim to output $\perp$ after the swapping, and erase the convincing slot in $B$. Then, undo will not be able to locate $D = B$ anymore.

**Claim 3.** *In an execution of* Sim *with a* bad *random tape, there are $k$ minimal containing blocks.*

*Proof.* As observed earlier, on a bad tape there will be $k$ convincing slots of the main session on the main thread (in order to reach END). We merely need to show that for each slot, its respective minimal containing block is distinct. Suppose that two slots share the same minimal containing block of length $t$. Since slots on the same thread are disjoint, we reach a contradiction as one of the slots must be properly contained in one of the two smaller blocks of size $t/2$. ∎

**Claim 4.** *Consider an execution of* Sim *with a bad random tape $\tau$. If there are $k'$ minimal rewinding block, then there are $k' - 2\log T$ composable blocks.*

*Proof.* Let $B$ be a minimal rewinding block that does not contain START or END. Since START (or END) can only be in at most $\log T$ different blocks on the main thread (since that is the recursion depth), we conclude that there are at least $k - 2\log T$ such blocks. It remains to show that $B$ is a composable block. Let $B'$ be the sibling of $B$.

The main block condition of composable blocks follows directly, while the signature condition on the main thread actually holds for the whole simulation from START to END, since $\tau$ is a bad random tape. Thus, we only need to show that the sibling condition is satisfied, i.e. $B'$ does not contain END. Assume the contrary that $B'$ does contain END. Since $B$ and $B'$ are siblings with a common starting point and $B$ contains a slot of the main session, $B'$ must contain that same slot in a convincing manner in order to reach END. On the other hand, $B$ does not contain END. Thus $B'$ will be executed before the main thread reaches END (if at all), and this convincing slot will allow Sim to extract the witness of the main session by the same argument in Claim 1. This contradicts the fact that $\tau$ is a bad tape. ∎

### 3.4.3 Concluding the Proof

We first show that Sim gets stuck with negligible probability, and then use it in Claim 7 to conclude that the output distribution of $S^{V^*}$ is computationally (respectively statistically) indistinguishable from the real view of $V^*$.

**Claim 5.** Sim *encounters rewinding failures with negligible probability.*

*Proof.* As mentioned before, since there are only polynomially many sessions and threads, it suffices to show that the probability of the simulator getting stuck on any fixed thread and session is negligible. The union bound then shows that Sim overall gets stuck with negligible probability

For any fixed thread and session, combining Claim 2, 3 and 4 shows that a random tape is bad with probability at most

$$\frac{1}{2^{k-2\log T}}$$

This is negligible in $n$ since $T$ is polynomial in $n$ and $k = \omega(\log n)$. ∎

**Claim 6.** Sim *encounters special-sound failures with negligible probability.*

*Proof.* Suppose for the sake of contradiction that Sim encounters special-sound failures with non-negligible probability. Consider an unbounded adversary that forwards a random pair of special-sound transcripts in an execution of Sim to an outside honest verifier of the special sound proof.

Although Sim may decide to rewind itself while the adversary is forwarding a partial transcript to the outside honest verifier, the adversary can continue the partial transcript by itself in exponential time[8]. This contradicts the statistical special-soundness property. ∎

**Claim 7.** *If the argument of knowledge in Stage 2 is $\mathcal{WI}$ (resp. statistical $\mathcal{WI}$), then the ensembles $\{View^P_{V^*}(x,z)\}_{x\in L, z\in\{0,1\}^*}$ and $\{\mathsf{Sim}(x,z)\}_{x\in L, z\in\{0,1\}^*}$ are computationally (resp. statistically) indistinguishable over $x \in L$.*

*Proof.* We consider polynomially many intermediate hybrids $\mathsf{Sim}_i$, $0 \leq i \leq m$, that receive the real witnesses to the statements $x_1, \ldots, x_m$. $\mathsf{Sim}_i$ proceeds as Sim until the $i^{\text{th}}$ Stage 2 proof on the output thread, after which $\mathsf{Sim}_i$ continues in a straight line simulation with $V^*$ using the real witnesses for Stage 2 proofs. $\mathsf{Sim}_i$ will output $\perp$, however, should Sim encounter a rewinding or special-sound failure during the $i^{\text{th}}$ proof. Clearly, $\mathsf{Sim}_0$ generates $View^P_{V^*}(x,z)$ and $\mathsf{Sim}_m$ generates $\mathsf{Sim}(x,z)$. Thus it is enough to show that for all $i$, the output of $\mathsf{Sim}_i$ and $\mathsf{Sim}_{i+1}$, are computationally (resp. statistically) indistinguishable.

We introduce yet another hybrid $\mathsf{Sim}'_i$ that proceeds as $\mathsf{Sim}_i$ except it utilizes the extracted fake witness for the $i^{\text{th}}$ proof. By Claim 5 and 6, $\mathsf{Sim}'_i$ is statistically close to $\mathsf{Sim}_{i+1}$ (since $\mathsf{Sim}_{i+1}$ may output $\perp$ with at most a negligibly increase probability between the $i^{\text{th}}$ and $i+1^{\text{st}}$ proofs). On the other hand, $\mathsf{Sim}_i$ and $\mathsf{Sim}_{i+1}$ differ only in the $i^{\text{th}}$ proof, and is therefore computationally (resp. statistically) indistinguishable by the $\mathcal{WI}$ property of the Stage 2 proof. Thus, the output of $\mathsf{Sim}_i$ and $\mathsf{Sim}_{i+1}$ are indeed computationally (resp. statistically) indistinguishable. ∎

Claim 7 completes the proof of Theorem 1 and 3.

*Remark.* Since we have shown that our lazy simulator is a concurrent zero-knowledge simulator, it follows directly that the KP simulator is also a concurrent zero-knowledge simulator: because the KP simulator receives more information than the lazy simulator at any point during the simulation (i.e. a bigger history repository $h$), the probability that the KP simulator outputs $\perp$ is no more than the probability that the lazy simulator outputs $\perp$. Thus, the same argument presented in Claim 7 can be applied also to the KP simulator.

# 4 Concurrent Zero-Knowledge Proofs

## 4.1 Protocol for Concurrent Zero-Knowledge Proofs

Our protocol for concurrent $\mathcal{ZK}$ proofs, ConcZKProof (depicted in Figure 5), is based on the protocol of Goldreich and Kahan [GK] with modifications introduced by [MP, PRS, MOSV]. The following primitives are used in ConcZKProof:

**Computational Special-Sound Proofs [CDS]:** A $k$-round interactive proof for language $L \in \mathcal{NP}$ with witness relation $R_L$ and security parameter $n$ is computational special-sound (SS) with respect to $R_L$ if the following holds: There exists a deterministic polynomial time procedure that can extract a witness with overwhelming probability in $n$, given a $k$-2-message prefix of the protocol ($\vec{\alpha}$) and two independent accepting completions of this prefix (($\vec{\alpha}, \beta, \gamma$) and ($\vec{\alpha}, \beta', \gamma'$)) that are generated by a computationally bounded prover and an honest verifier.

[8] In Section 4.2, a more involved proof is needed since only computational special-soundness is guaranteed

4-round *statistical* $\mathcal{WI}$ and *computational* special-sound proofs of knowledge for $\mathcal{NP}$ can be based on 2-round statistically hiding commitments.

**Commitments (see e.g. [G]):** An (interactive) commitment scheme consists of a sender $S$, a receiver $R$, and a verification algorithm Ver. A message $m$ is given as private input to $S$, and after the interaction $R$ outputs a commitment string $c$ and $S$ outputs a decommitment pair $(m, d)$. Given $(c, m, d)$, Ver accepts if $(m, d)$ is a valid decommitment of $c$, and rejects otherwise. The commitment scheme is correct if Ver accepts all $(c, m, d)$ triples generated by the scheme.

The commitment scheme is binding (resp. statistically binding) if for all efficient (resp. unbounded) senders $S^*$ interacting with the honest receiver $R$, the probability of generating a commitment $c$ and two valid decommitments of $c$ to two distinct messages is negligible.

The commitment scheme is hiding (resp. statistically hiding) if for all efficient (resp. unbounded) receivers $R^*$ interacting with the honest sender $S$, the probability that the $R^*$ can distinguish between the commitment of two messages is negligible.

**Public-coin Committed Verifier $\mathcal{ZK}$ Proofs (CVZK) [CDS, MOSV]:** A committed-verifier $V_m$, where $m = (m_1, \ldots, m_k)$, is a deterministic verifier that always sends $m_i$ as its $i^{\text{th}}$ message. A public-coin interactive proof is computational (resp. statistical) committed-verifier zero knowledge (CVZK) if there exists a probabilistic polynomial-time simulator $S$ such that for all committed verifier $V_m$, the output of $S(m)$ is computationally (resp. statistically) indistinguishable from the view of $V_m$ in a real execution of th protocol. The GMW graph 3-coloring [GMW] and the Blum Hamiltonicity [Blum] protocols are examples of 4-round public-coin computational CVZK proofs for $\mathcal{NP}$ with efficient provers that can be based on one-way functions. As noted by [MOSV], any language that has a public-coin honest verifier zero-knowledge proof also has a CVZK proof with one additional round.

The completeness and the soundness of the protocol follows directly from the proof of Goldreich and Kahan [GK]; in fact, the protocol is an instantiation of theirs. Intuitively, the prover cannot cheat since the initial commitment is statistically hiding. Note that ConcZKArg and ConcZKProof share a very similar structure. In both cases, Stage Init is used to fix the "fake witnesses" of the session. This is followed by $k$ slots of special sound proofs in Stage 1 that gives the simulator a chance to extract a "fake witness". The final proof or argument in Stage 2 can be completed by using either a real witness of the language or a "fake witness" extracted in Stage 1.

### 4.2 Proof of Theorem 4

In this section we describe a concurrent zero-knowledge simulator for ConcZKProof. Actually, we will continue to use Sim as our simulator with a small modification: During Stage 2, assuming that Sim has been able to extract $\bar{r}$ from the commitment in Stage Init, it will generate the Stage 2 proof using the CVZK property. Sim may now output $\perp$ for three different reasons:

1. Sim is did not obtain two matching special sound proof transcripts in Stage 1. Call this a *rewinding failure.*

2. Sim could not extract a witness even after obtaining two matching special sound proofs in Stage 1. Call this a *special-sound failure.*

PROTOCOL CONCZKPROOF

**Common Input:** an instance $x$ of a language $L$ with witness relation $R_L$.

**Auxiliary Input for Prover:** a witness $w$, such that $(x, w) \in R_L(x)$.

**Stage Init:**

V uniformly chooses $\overline{r} = r_1, \ldots, r_\ell \in \{0,1\}^n$, $s_1, \ldots, s_\ell \in \{0,1\}^{poly(n)}$, where $\ell = \ell(n)$ is determined in Stage 2.

V $\leftrightarrow$ P: Compute $c_i = \mathsf{Com}(r_i, s_i)$ in parallel for $1 \leq i \leq \ell$, where $\mathsf{Com}$ is a statistically hiding commitment that may be interactive.

V $\leftrightarrow$ P: Interactively exchange in parallel all but the last two messages of $k\ell$ copies ($k$ copies for each $c_i$) of a statistically $\mathcal{WI}$ and computationally special-sound proof of knowledge (CSS-POK) of the statement:

there exists values $\overline{r}', s'$ s.t. $c_i = \mathsf{Com}(\overline{r}'; s')$

The proof of knowledge is with respect to the witness relation $R'_L(c) = \{(v, s) : c = \mathsf{Com}(v; s)\}$.

We say the protocol has reached START (of Stage 1) if all messages in Stage Init are exchanged.

**Stage 1:**

For $1 \leq j \leq k$ do

P $\rightarrow$ V: The second last messages of the $j^{\text{th}}$ batch of $\mathcal{WI}$ CSS-POKs (one for each $1 \leq i \leq \ell$).

V $\rightarrow$ P: The last messages of the $j^{\text{th}}$ batch of $\mathcal{WI}$ CSS-POKs (one for each $1 \leq i \leq \ell$).

We say the protocol has reached END (of Stage 1) if all $k$ CSS-POKs are accepted.

**Stage 2:**

P $\leftrightarrow$ V: $P$ proves to $V$ that $x \in L$ using a $\ell(n)$-round public-coin computational CVZK proof with efficient provers. During the proof, the verifier decommits to $r_i$ as its $i^{\text{th}}$ message.

Figure 5: Concurrent $\mathcal{ZK}$ Proof for $\mathcal{NP}$ with parameter $k$.

3. During Stage 2, $V^*$ decommits the initial commitment $\mathsf{Com}$ to a value $\overline{r}$ different from the value extracted by $\mathsf{Sim}$. Call this a *binding failure* (since $V^*$ has successfully opened the initial commitment to two values).

Intuitively, special-sound failures and binding failures should occur with negligible probabilities as well since $V^*$ is computationally bounded. We give the formal proofs below:

**Claim 8.** $\mathsf{Sim}$ *encounters rewinding failures with negligible probability.*

*Proof.* This can be shown in the same manner as Claim 5 using the analysis in Section 3.4. ∎

**Claim 9.** $\mathsf{Sim}$ *encounters special-sound failures with negligible probability.*

*Proof.* Suppose for the sake of contradiction that $\mathsf{Sim}$ does encounter special-sound failures with non-negligible probability. Then for some polynomial $p$ and infinitely many $n$, there exist initial transcripts $\tau = \tau_n$ such that immediately after $\tau$, $\mathsf{Sim}$ will produce a prefix of the computational special-sound proof and two independent completions of the transcript that prevents witness extraction with probability at least $1/p(n)$.

We may consider the following experiment instead: First, choose a random thread of $\mathsf{Sim}$, and execute $\mathsf{Sim}$ continuing from transcript $\tau$ until a prefix $\tau_1$ of a Stage 1 special-sound proof appears on the chosen thread; next, pick two random threads of the simulation to obtain a pair of special-sound transcripts following $\tau_1$. Let $\Pr^{SS}(\mathsf{Sim}) = \Pr^{SS}(\mathsf{Sim}, n)$ denote the probability that the pair of special-sound transcripts generated in the experiment prevents witness extraction. Since there are only polynomially many threads, there exists a polynomial $q$ such that $\Pr^{SS}(\mathsf{Sim}) > 1/q(n)$

This does not directly contradict the computational special-sound property, however, since $\mathsf{Sim}$, acting as an adversarial prover for the special-sound proof and interacting with an honest verifier, may need to rewind the honest verifier during its execution. Instead, we consider a modified simulator $\mathsf{Sim}'$ that receives the real witnesses to the statements $x_1, \ldots, x_m \in L$; $\mathsf{Sim}'$ will continue from the prefix $\tau$ in a straight-line simulation (on the chosen threads) using the real witnesses for any Stage 2 proofs. If $\Pr^{SS}(\mathsf{Sim}')$ (defined similarly) is also non-negligible, then $\mathsf{Sim}'$ can be used to contradict the computational special-sound property.

To prove it, we consider hybrids $\mathsf{Sim}_i$, $0 \leq i \leq m$, defined as follows. $\mathsf{Sim}_i$ proceeds as $\mathsf{Sim}$ until the $i^{\text{th}}$ Stage 2 proof (on the chosen thread), after which $\mathsf{Sim}_i$ continues as $\mathsf{Sim}'$ (i.e. does a straight-line simulation of the chosen thread), but will output $\perp$ should a rewinding, special-sound or binding failure occur during the $i^{\text{th}}$ proof. Clearly, $\mathsf{Sim} = \mathsf{Sim}_m$, and $\mathsf{Sim}' = \mathsf{Sim}_0$. Suppose for the sake of contradiction that $\Pr^{SS}(\mathsf{Sim}')$ is negligible. Since $m$ is polynomial in $n$, there exists some $i$ such that $\Pr^{SS}(\mathsf{Sim}_{i+1}) - \Pr^{SS}(\mathsf{Sim}_i) \geq 1/(2p(n)m(n))$. Let us introduce one more hybrid simulator, $\mathsf{Sim}'_i$, that proceed as $\mathsf{Sim}_i$ but uses the extracted witness from Stage 1 proofs to provide the $i^{\text{th}}$ Stage 2 proof. $\Pr^{SS}(\mathsf{Sim}'_i) \geq \Pr^{SS}(\mathsf{Sim}_{i+1})$ since $\mathsf{Sim}_{i+1}$ and $\mathsf{Sim}'_i$ are identical except that $\mathsf{Sim}_{i+1}$ may abort more often during rewinds between the $i^{\text{th}}$ and $i+1^{\text{st}}$ Stage 2 proof. This gives us

$$\Pr^{SS}(\mathsf{Sim}'_i) - \Pr^{SS}(\mathsf{Sim}_i) \geq \frac{1}{2p(n)m(n)}$$

But $\mathsf{Sim}_i$ and $\mathsf{Sim}'_i$ differ only in the $i^{\text{th}}$ proof. This contradicts the computational-CVZK property of the Stage 2 proof. ∎

**Remark:** In the proof of Claim 9 we required that $L \in \mathcal{NP}$ and that the Stage 2 CVZK proof has an efficient prover to construct the hybrids $\mathsf{Sim}_i$; these two properties are not used elsewhere in the analysis of ConcZKProof. Looking forward, we will be able to relax these assumptions in Section 5 when the verifier uses statistically binding commitments.

**Claim 10.** Sim *encounters binding failures with negligible probability.*

*Proof.* This claim can be shown in the same manner as Claim 9. On a high level, should Sim encounter binding failures with non-negligible probability, we can use Sim to produce two openings of the initial commitment with non-negligible probability, and therefore contradict the computational binding property of Com. ∎

**Claim 11.** *Let $(P, V)$ be defined as in ConcZKProof. Then $\{View^P_{V^*}(x, z)\}_{x\in L, z\in\{0,1\}^*}$ and $\{\mathsf{Sim}(x, z)\}_{x\in L, z\in\{0,1\}^*}$ are computationally indistinguishable over $x \in L$.*

*Proof.* Claim 8, 9 and 10 show that Sim outputs $\perp$ with negligible probability. Thus, we can follow the same proof as in 7 (using the CVZK property instead of the $\mathcal{WI}$ property) to conclude that the output of Sim is indistinguishable from the view of a verifier in a real execution. ∎

# 5 Unconditional Characterizations of Concurrent Zero Knowledge

Let $\mathcal{SZKP}$ (resp. $\mathcal{SZKA}$) denote the set of languages that have a statistical $\mathcal{ZK}$ proofs (resp. arguments) and $\mathcal{CZKP}$ (resp. $\mathcal{CZKA}$) denote the set of languages having computational $\mathcal{ZK}$ proofs (resp. arguments). We will utilize unconditional constructions of *instance-based commitments* [BMO, OV07, OV08] to unconditionally characterize the languages that have concurrent $\mathcal{ZK}$ protocols.

Informally, an *instance-based commitment* scheme for a language $L$ is a family of efficient standard commitment schemes $\{\mathsf{Com}_x\}_x$ where both the sender and the receiver receives the instance $x$ in protocol $\mathsf{Com}_x$. The commitment satisfies the following two properties:

1. If $x \in L$, the commitment is hiding
2. If $x \notin L$, the commitment is binding.

As usual, the binding and hiding properties can be statistical or computational. From now on, when we describe the binding (resp. hiding) property of an instance-based commitment, we implicitly require the property to hold only for the instances of the language (resp. the complement of the language). For an immediate reference, we provide formal definitions of instance-based commitments in Appendix B.

On a high-level, we implement the primitives in ConcZKProof with instance-based commitments to achieve unconditional results. For the $\mathcal{ZK}$ property, we need a commitment scheme that is hiding whenever $x \in L$ (e.g. to implement GMW in Stage 2). For the soundness property, however, we need a commitment scheme that is hiding whenever $x \notin L$ for the Stage Init of the protocol (this is opposite from the definition of instance-based commitments). Therefore, to construct a concurrent $\mathcal{ZK}$ protocol, we require instance-based commitment schemes for both $L$ and $\bar{L}$.

For the remainder of this section, let $t(n)$ be an upper bound on the round complexity of any statistically hiding commitment based on one-way functions. [HR, NOV] showed that $t(n)$ is a

polynomial. The following theorem shows that languages with zero-knowledge protocols do have the necessary instance-based commitments:

**Theorem 7.** *If $L \in \mathcal{SZKP}$, then we have an $O(1)$-round, statistically hiding and statistically binding instance-based commitment for $L$ and $\bar{L}$. Similar statements also hold for the classes $\mathcal{CZKP}$, $\mathcal{SZKA}$ and $\mathcal{CZKA}$ as shown in the table below:*

| | Instance-based commitment for $L$ | | | Instance-based commitment for $\bar{L}$ | | |
|---|---|---|---|---|---|---|
| | rounds | hiding | binding | rounds | hiding | binding |
| $L \in \mathcal{SZKP}$ | $O(1)$ | stat. | stat.* | $O(1)$ | stat. | stat.* |
| $L \in \mathcal{CZKP}$ | $O(1)$ | comp. | stat.* | $O(t(n))$ | stat. | comp. |
| $L \in \mathcal{SZKA} \cap \mathcal{NP}$ | $O(t(n))$ | stat. | comp.* | $O(1)$ | comp. | stat. |
| $L \in \mathcal{CZKA} \cap \mathcal{NP}$ | $O(t(n))$ | comp. | comp.* | $O(t(n))$ | comp. | comp. |

The proof of Theorem 7 combines results from [OV07, OV08, Oka] and is found in Appendix C. Theorem 7 intuitively says the following: Let $A$ and $B$ be the descriptions "statistical" or "computational". Let us write $(A, B)$-$\mathcal{ZK}$ to denote zero-knowledge with $A$-indistinguishability and $B$-soundness, and $(A, B)$-Com to denote $A$-hiding and $B$-binding instance-based commitments. The theorem states that

$$L \in (A, B)\text{-}\mathcal{ZK} \Rightarrow L \text{ has an } (A, B)\text{-Com and } \bar{L} \text{ has a } (B, A)\text{-Com}$$

We now proceed with our unconditional characterizations of languages with concurrent zero-knowledge proofs:

**Theorem 8.** *Every language $L \in \mathcal{SZKP}$ has a $\omega(\log n)$-round statistical black-box concurrent $\mathcal{ZK}$ proof.*

*Proof.* We begin with the protocol ConcZKProof and make some modifications. First, we use instance-based commitments (from Theorem 7) for $\bar{L}$ to implement the initial commitment and the proofs of knowledge in Stage Init and Stage 1. We also replace Stage 2 with a constant-round public-coin statistical CVZK protocol for $L$; such a protocol exists for all $L \in \mathcal{SZKP}$ as shown in [OV07, MOSV]. Since the instance-based commitments are constant-round, the modified ConcZKProof continues to have round-complexity $\omega(\log n)$.

Completeness of the modified ConcZKProof holds trivially, and soundness holds since the instance-based commitment used in Stage Init is statistically hiding when $x \notin L$ (i.e. when $x \in \bar{L}$). To establish the zero-knowledge property, we use the same simulator Sim as before. We start by bounding the failure probability of Sim:

**Rewinding failures:** Sim encounters rewinding failures with negligible probability using the same analysis as in Claim 5.

**Special-sound and binding failures:** Since the proofs of knowledge are statistically special-sound and the initial commitment is statistically binding, both failures must occur with negligible probability using an analysis similar to that of Claim 6. Note that unlike the analysis of Claim 9, we do not require the language $L \in \mathcal{NP}$.

*These results were stated already in [OV07, OV08].

Having established that Sim fails with negligible probability, we can now apply the analysis of Claim 7 (using the CVZK property instead of the $\mathcal{WI}$ property) to establish the statistical zero-knowledge property. ∎

Next, we turn to the class of computational zero-knowledge proofs:

**Theorem 9.** *Every language $L \in \mathcal{CZKP} \cap \mathcal{NP}$ has a $\omega(t(n)\log n)$-round computational black-box concurrent $\mathcal{ZK}$ proof.*

*Proof.* Again, we begin with the protocol ConcZKProof and use instance-based commitments (from Theorem 7) for $\bar{L}$ to implement the initial commitment and the proofs of knowledge in Stage Init and Stage 1. We also use the instance-based commitments for $L$ to implement the CVZK proof in Stage 2; this is possible for languages in $\mathcal{NP}$, using the GMW 3-coloring protocol [GMW] for example. Since the instance-based commitment for $\bar{L}$ has round complexity $O(t(n))$, the modified ConcZKProof now has round-complexity $\omega(t(n)\log n)$. Completeness and soundness holds similarly as in the previous theorem. The computational zero-knowledge property is more involved though. Since $L \in \mathcal{CZKP}$, the instance-based commitment for $\bar{L}$ is only computationally binding. Therefore, the analysis of Sim will follow that of Claims 8, 9, 10, and 11. In particular, because the analysis of Claim 9 requires the construction of a hybrid simulator that receives a witness to each instance of $x \in L$, this theorem only holds of languages $L \in \mathcal{NP}$. ∎

We also consider the case of zero-knowledge arguments:

**Theorem 10.** *Every language $L \in \mathcal{SZKA} \cap \mathcal{NP}$ has a $O(t(n))+\omega(\log n)$-round statistical black-box concurrent $\mathcal{ZK}$ argument. Similarly, every language $L \in \mathcal{CZKA} \cap \mathcal{NP}$ has a $\omega(t(n)\log n)$-round computational black-box concurrent $\mathcal{ZK}$ argument.*

*Proof.* We will actually continue to use the protocol ConcZKProof instead of ConcZKArg, since we do not assume the existence of one-way functions. As before, we use instance-based commitments for $\bar{L}$ to implement Stage Init and Stage 1 of ConcZKProof, and use instance-based commitments for $L$ to implement Stage 2.

In order for the protocol to remain sound even though the verifier's commitments are only computationally hiding, the verifier no longer reveals its commitments in Stage 2. Instead, the verifier gives a zero-knowledge proof of knowledge in Stage 2 to show that its Stage 2 messages are valid openings of its Stage Init commitments (and does so using instance-based commitments for $\bar{L}$). See [G] for details on this technique.

Completeness and zero-knowledge again follows from the analysis of ConcZKProof (appropriately modified depending on whether statistical or computational zero-knowledge is required). ∎

**Remark:** Theorem 10 can be extended to give a compiler from $s(n)$-round, public-coin, statistical HVZK arguments into $s(n) + \omega(\log n)$-round statistical black-box concurrent $\mathcal{ZK}$ arguments for languages in $\mathcal{NP}$. Since $\mathcal{NP}$ languages with statistical HVZK arguments are in $\mathcal{SZKA}$ [OV07, Theorem 3.4], we can apply Theorem 7 to obtain instance-based commitments for $\bar{L}$. Meanwhile, we can transform the HVZK protocol into a CVZK protocol (with a constant round increase, [MOSV]) for use in Stage 2 (similar to Theorem 8). Additionally, the compiler gives stand-alone $\mathcal{SZKA}$ protocols without the $\omega(\log n)$ overhead by setting $k = 1$ in ConcZKProof (so that Stage 1 is constant round).

## 6 Acknowledgements

We are grateful to Manoj Prabhakaran, Alon Rosen, Amit Sahai and Salil Vadhan for helpful discussions.

## References

[Blum] M. Blum. How to prove a theorem so no one else can claim it. In *Proc. of the International Congress of Mathematicians*, Berkeley, California, USA, pages 1444–1451, 1986.

[BG] M. Bellare and O. Goldreich. On defining proofs of knowledge. In *CRYPTO92*, Springer LNCS 740, pages 390–420.

[BGGHKMR] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Håstad, J. Kilian, S. Micali, P. Rogaway. Everything provable is provable in zero-knowledge. In *Proc. CRYPTO '88*, pages 37–56, 1988.

[BMO] M. Bellare, S. Micali and R. Ostrovsky. Perfect zero-knowledge in constant rounds. In *Proceedings of th 22nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 482–493, 1990.

[BPS] B. Barak, M. Prabhakaran and A. Sahai. Concurrent non-malleable zero knowledge. In *47th FOCS*, pages 345–354, 2006.

[CDS] R. Cramer, I. Damgård and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Crypto94*, Springer LNCS 839, pages 174–187, 1994.

[CGGM] R. Canetti, O. Goldreich, S. Goldwasser and S. Micali. Resettable zero-knowledge. In *32nd STOC*, pages 235–244, 2000.

[CKPR] R. Canetti, J. Kilian, E. Petrank and A. Rosen. Black-box concurrent zero-knowledge requires (almost) logarithmically many rounds. In *SIAM Jour. on Computing*, Vol. 32(1), pages 1–47, 2002.

[D00] I. Damgård. Efficient concurrent zero-knowledge in the auxiliary string model. In *EuroCrypt2000*, LNCS 1807, pages 418–430, 2000.

[DDN] D. Dolev, C. Dwork and M. Naor. Non-malleable cryptography. In *SIAM Journal on Computing*, Vol. 30(2), pages 391–437, 2000.

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

[DS] C. Dwork and A. Sahai. Concurrent zero-knowledge: reducing the need for timing constraints. In *Crypto98*, Springer LNCS 1462 , pages 442–457, 1998.

[ESY] S. Even, A. L. Selman and Y. Yacobi. The complexity of promise problems with applications to public-key cryptography. *Information and Control*, 61(2):159–173, May 1984.

[G] O. Goldreich. Foundations of Cryptography. Cambridge University Press, 2001.

[FS] U. Feige and A. Shamir. Witness indistinguishable and witness hiding proofs. In *22nd FOCS*, pages 416–426, 1990.

[GK] O. Goldreich and A. Kahan. How to construct constant-round zero-knowledge proof systems for NP. In *Journal of Cryptology*, Vol. 9(3), pages 167–189, 1996.

[GMOS] V. Goyal, R. Moriarty, R. Ostrovsky, A. Sahai. Concurrent statistical zero-knowledge arguments for NP from one way functions. In *ASIACRYPT*, pages 444-459, 2007.

[GMR] S. Goldwasser, S. Micali and C. Rackoff. The knowledge complexity of interactive proof systems. In *SIAM Jour. on Computing*, Vol. 18(1), pages 186–208, 1989.

[GMW] O. Goldreich, S. Micali and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[HKKL] C. Hazay, J. Katz, C.Y. Koo, and Y. Lindell. Concurrently-secure blind signatures without random oracles or setup assumptions. In *Theory of Cryptography*, Springer LNCS 4392, pages 323–341, 2007.

[HR] I. Haitner and O. Reingold. Statistically-hiding commitment from any one-way function. In *39th STOC*, pages 1–10.

[KP] J. Kilian and E. Petrank. Concurrent and resettable zero-knowledge in poly-logarithmic rounds. In *33rd STOC*, pages 560–569, 2001.

[KPR] J. Kilian, E. Petrank and C. Rackoff. Lower bounds for zero-knowledge on the internet. In *39th FOCS*, pages 484–492, 1998.

[MP] S. Micali and R. Pass. Local zero knowledge. In *STOC'06*.

[MOSV] D. Micciancio, S.J. Ong, A. Sahai, S. Vadhan. Concurrent zero knowledge without complexity assumptions. In *Theory of Cryptography*, Springer LNCS 3876, pages 1–20, 2006.

[Naor] M. Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[NOV] M. Nguyen, S.J. Ong, S. Vadhan. Statistical zero-knowledge arguments for NP from any one-way function. In *47th FOCS*, pages 3–14, 2006.

[Oka] T. Okamoto. On relationships between statistical zero-knowledge proofs. *Journal of Computer and System Sciences*, 60(1):47–108, 2000.

[OV07] S.J. Ong, S. Vadhan. Zero knowledge and soundness are symmetric. In *Proc. EUROCRYPT 2007*, pages 187–209, 2007.

[OV08] S.J. Ong, S. Vadhan. An equivalence between zero knowledge and commitments. To appear in *Theory of Cryptography*, 2008.

[P06] R. Pass. A precise computational approach to knowledge. PhD thesis, MIT, 2006.

[PPSTV] O. Pandey, R. Pass, A. Sahai, W.D. Tseng and M. Venkitasubramaniam. Concurrent precise zero knowledge. To appear in *EuroCrypt2008*.

[PRS] M. Prabhakaran, A. Rosen and A. Sahai. Concurrent zero-Knowledge with logarithmic round complexity. In *43rd FOCS*, pages 366–375, 2002.

[PS] M. Prabhakaran and A. Sahai. Concurrent zero knowledge proofs with logarithmic round-complexity. In *Cryptology ePrint Archive*, report 2002/055.

[PV] R. Pass, M. Venkitasubramaniam. On constant-round concurrent zero-knowledge. To appear in *Theory of Cryptography*, 2008.

[R00] A. Rosen. A note on the round-complexity of concurrent zero-knowledge. In *Crypto2000*, Springer LNCS 1880, pages 451–468, 2000.

[RK] R. Richardson and J. Kilian. On the concurrent composition of zero-knowledge proofs. In *EuroCrypt99*, Springer LNCS 1592, pages 415–431, 1999.

[R04] A. Rosen. The round-complexity of black-box concurrent zero-knowledge. PhD thesis, Weizmann Institute of Science, Israel (2003).

# A The PRS Analysis [PRS]

The PRS approach of mapping bad random tapes to good random tapes is quite different from the approach taken in this paper. In this section, we briefly describe the tools used in the PRS analysis and discuss some subtleties in their analysis.

Given a bad random tape, the PRS analysis deals with *minimal rewinding intervals*, defined to be minimal blocks that contain a slot, without containing START or END.[9] Since minimal rewinding intervals are not "composable" when they overlap, the PRS analysis focuses on a (maximal) set of *disjoint* minimal rewinding intervals. To make up for lost intervals due to overlapping, the PRS analysis swaps each minimal rewinding interval not only with its sibling (as we do), but also with its "cousins". See Figure 6 for an illustration of cousins blocks. Note that a block may have many cousins (but only one sibling). Moreover, swapping a block with its sibling may require an exchange of random tape segments outside the two blocks, and therefore produce changes in the simulation outside of the cousins

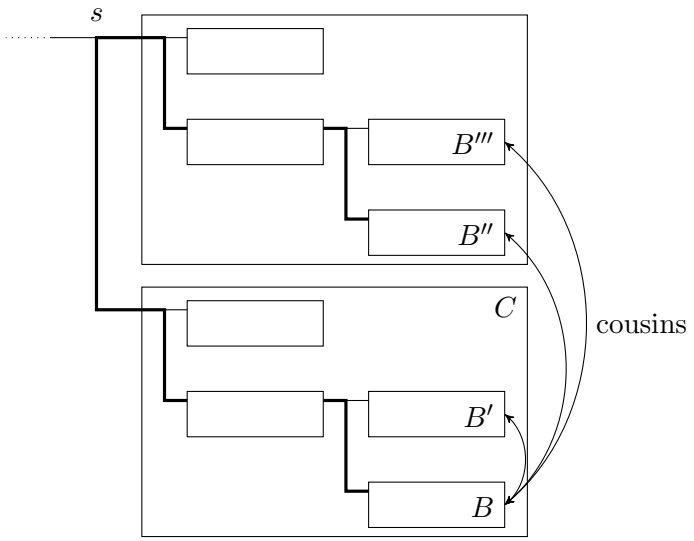



Figure 6: A pictorial representation of the original KP rewind schedule, extended from Figure 5 of [PRS]. We show how a rewinding interval $B$ is related to its sibling ($B'$), its parent ($C$), and its cousins ($B', B'', B'''$). To swap block $B$ with its cousin $B''$, one needs to exchange the randomness used on the two highlighted thread.

Next, the analysis needs to determine for each rewinding interval, how many cousins swaps will result in a new distinct random tape; this step is complicated because a large portion of the random tape maybe shuffled to perform a cousin swap, destroying other potential rewinding intervals. Each rewinding interval is thus assigned a weight corresponding to the number of available cousins. Finally, a rather elaborate analysis is used to lower-bound the sum of weights over the chosen (maximal) subset of disjoint rewinding intervals. (Recall that, in contrast, our analysis is local—we are only required to show that a *single* swap of a block with its sibling results in *one* new random tape.)

[9] Here we adopt some of our terminologies to explain the PRS analysis.

## B Instance-based Commitments

We recall the definitions of instance based commitments from [OV07].

**Definition 4** ([OV07, Definition 2.7]). *An* instance-dependent *commitment scheme is a family* $\{\mathsf{Com}\}_{x\in\{0,1\}^*}$ *with the following properties:*

1. $\mathsf{Com}_x = \langle C_x, R_x\rangle$ *consists of a commit and a reveal stage. In both stages, the committer* $C$ *and the receiver* $R$ *receive instance* $x$ *as common input.*
2. *At the beginning of the commit stage, sender* $C$ *receives a private input* $b \in \{0,1\}$*. At the end of the commit stage, both the sender and receiver output a commitment* $c$*.*
3. *In the reveal stage,* $C$ *sends a pair* $(b, d)$*, where* $d$ *is interpreted as the decommitment string for bit* $b$*.* $R$ *accepts or rejects based on* $x, b, d, c$*.*
4. $C$ *and* $R$ *are computable in polynomial time (in* $|x|$*).*
5. *For every* $x \in \{0,1\}^*$*,* $R$ *always accepts with probability 1 if both* $C$ *and* $R$ *follow their strategy honestly.*

**Definition 5** ([OV07, Definition 2.8]). *Instance-based commitment scheme* $\mathsf{Com}_x = \langle C_x, R_x\rangle$ *is* statistically (resp. computationally) hiding *on* $I \subseteq \{0,1\}^*$ *if for every (resp. nonuniform PPT)* $R^*$*, the ensembles* $\{View_{R^*}(C_x(0), R^*)\}_{x\in I}$ *and* $\{View_{R^*}(C_x(1), R^*)\}_{x\in I}$ *are statistically (resp. computationally) indistinguishable, where* $\{View_{R^*}(C_x(b), R^*)\}_{x\in I}$ *denotes the view of* $R^*$ *in the commit stage interacting with* $C_x$*.*

**Definition 6** ([OV07, Definition 2.9]). *Instance-based commitment scheme* $\mathsf{Com}_x = \langle C_x, R_x\rangle$ *is* statistically (resp. computationally) binding *on* $I \subseteq \{0,1\}^*$ *if for every (resp. nonuniform PPT)* $S^*$*, there exists a negligible function* $\varepsilon(\cdot)$ *such that for all* $x \in I$*, the adversarial sender* $S^*$ *succeeds in the following game with probability at most* $\varepsilon(|x|)$*.*

> $S^*$ *interacts with* $R_x$ *in the commit stage obtaining commitment* $c$*. Then* $S^*$ *outputs pairs* $(0, d_0)$ *and* $(1, d_1)$*, and succeeds if in the reveal stage,* $R_x(0, d_0, c) = R_x(1, d_1, c) =$ $\mathsf{accept}$.

Finally, for a language $L$, we say that instance-based commitment $\mathsf{Com}_x$ for $\Pi$ is hiding and binding if $\mathsf{Com}_x$ is hiding on $L$ and binding on $\bar{L}$.

## C Proof of Theorem 7

We first recall the characterization of $\mathcal{ZK}$ languages and instance-based commitments by Ong and Vadhan in [OV07]. We start with two definitions:

**Definition 7** (Promise Problems [ESY]). *A* promise problem *is specified by two disjoint sets of strings* $\Pi = (\Pi_Y, \Pi_N)$*, where* $\Pi_Y$ *is the set of YES instances and* $\Pi_N$ *is the set of NO instances. The complement of a promise problem is the promise problem* $\bar{\Pi} = (\Pi_N, \Pi_Y)$*.*

**Definition 8** ([OV07, Definition 1.3]). *A problem $\Pi = (\Pi_Y, \Pi_N)$ satisfies the SZKP-OWF CONDITION if there exists a set of instances $I \subseteq \Pi_Y \cup \Pi_N$ such that the following two conditions hold.*

- *The promise problem $(\Pi_Y \backslash I, \Pi_N \backslash I)$ is in SZKP.*
- *There exists a polynomial-time computable function $f_x : \{0,1\}^{n(|x|)} \to \{0,1\}^{m(|x|)}$, with $n(\cdot)$ and $m(\cdot)$ being polynomials and instance $x$ given as an auxiliary input such that for any instance $x \in I$, $f_x$ is a one-way function.*

[OV07] shows the following characterization of $\mathcal{ZK}$ languages:

**Claim 12** ([OV07, Theorem 1.4]). *Let $\Pi = (\Pi_Y, \Pi_N)$ be a promise problem in $\mathcal{IP}$ that satisfies the SZKP-OWF CONDITION. Let $I \subseteq (\Pi_Y, \Pi_N)$ be the set with the required properties stated in the SZKP-OWF CONDITION condition.*

1. *($\mathcal{SZKP}$) $\Pi$ has a statistical zero-knowledge proof system if and only if $\Pi$ satisfies the SZKP-OWF CONDITION without OWF instances, i.e. $I = \emptyset$*
2. *($\mathcal{CZKP}$) $\Pi$ has a computational zero-knowledge proof system if and only if $\Pi$ satisfies the SZKP-OWF CONDITION without OWF NO instance, i.e. $I \cap \Pi_N = \emptyset$*
3. *($\mathcal{SZKA}$) If $\Pi \in \mathcal{NP}$, $\Pi$ has a statistical zero-knowledge argument system if and only if $\Pi$ satisfies the SZKP-OWF CONDITION without OWF YES instance, i.e. $I \cap \Pi_Y = \emptyset$*
4. *($\mathcal{CZKA}$) If $\Pi \in \mathcal{NP}$, $\Pi$ has a computational zero-knowledge argument system if and only if $\Pi$ satisfies the SZKP-OWF CONDITION.*

[OV07, OV08] also shows that languages that satisfy the SZKP-OWF CONDITION condition have unconditional constructions of instance-based commitments. Recall that $t(n)$ is an upper bound on the round complexity of any statistically hiding commitment based on one-way functions.

**Claim 13** ([OV07, Lemma 3.13]). *Let $\Pi = (\Pi_Y, \Pi_N)$ be in $\mathcal{IP}$.*

- *If $\Pi$ satisfies the SZKP-OWF CONDITION without OWF NO instances (resp. without OWF instances), then it has an instance-based commitment scheme that is computationally (resp. statistically) hiding and statistically binding. Moreover, this scheme is public-coin and constant round.*
- *If $\Pi$ satisfies the SZKP-OWF CONDITION (resp. without OWF YES instance), then it has an instance-based commitment scheme that is computationally (resp. statistically) hiding and computationally binding. Moreover, this scheme is public-coin and has $O(t(n))$-rounds*[10].

We restate Theorem 7:

**Theorem 11** (Theorem 7 restated). *If $L \in \mathcal{SZKP}$, then we have an $O(1)$-round, statistically hiding and statistically binding instance-based commitment for $L$ and $\bar{L}$. Similar statements also hold for the classes $\mathcal{CZKP}$, $\mathcal{SZKA}$ and $\mathcal{CZKA}$ as shown in the table below:*

[10] This is not explicitly stated in in [OV07], but it is easy to see that it follows from their analysis

| | Instance-based commitment for $L$ | | | Instance-based commitment for $\bar{L}$ | | |
|---|---|---|---|---|---|---|
| | rounds | hiding | binding | rounds | hiding | binding |
| $L \in \mathcal{SZKP}$ | $O(1)$ | stat. | stat. | $O(1)$ | stat. | stat. |
| $L \in \mathcal{CZKP}$ | $O(1)$ | comp. | stat. | $O(t(n))$ | stat. | comp. |
| $L \in \mathcal{SZKA} \cap \mathcal{NP}$ | $O(t(n))$ | stat. | comp. | $O(1)$ | comp. | stat. |
| $L \in \mathcal{CZKA} \cap \mathcal{NP}$ | $O(t(n))$ | comp. | comp. | $O(t(n))$ | comp. | comp. |

*Proof of Theorem 7.* We will only show the theorem for the case $L \in \mathcal{CZKP}$; the other cases can be established in a very similar manner.

Let $\Pi = L$ (i.e. $\Pi_Y = L, \Pi_N = \bar{L}$). Since $\Pi \in \mathcal{CZKP}$, by Claim 12, $\Pi$ satisfies the SZKP-OWF CONDITION with $I \cap \Pi_N = \emptyset$. Claim 13 then immediately gives us the desired instance-based commitment for $\Pi = L$.

Next, consider $\bar{\Pi} = (\bar{\Pi}_Y = \Pi_N, \bar{\Pi}_N = \Pi_Y)$. Since $(\Pi_Y \setminus I, \Pi_N \setminus I) \in \mathcal{SZKP}$ ($\Pi$ satisfies the SZKP-OWF CONDITION), and since $\mathcal{SZKP}$ is closed under complement [Oka], we have $(\Pi_N \setminus I, \Pi_Y \setminus I) \in \mathcal{SZKP}$ as well. Hence, $\bar{\Pi}$ also satisfies the SZKP-OWF CONDITION with $I \cap \bar{\Pi}_Y = I \cap \Pi_N = \emptyset$. Applying Claim 13 again gives us the desired instance-based commitment for $\bar{\Pi} = \bar{L}$. ∎