The Knowledge Tightness of Parallel Zero-Knowledge

Kai-Min Chung* Rafael Pass† Wei-Lung Dustin Tseng‡

September 22, 2011

Abstract

We investigate the concrete security of black-box zero-knowledge protocols when composed in parallel. As our main result, we give essentially tight upper and lower bounds (up to logarithmic factors in the security parameter) on the following measure of security (closely related to knowledge tightness): the number of queries made by black-box simulators when zero-knowledge protocols are composed in parallel. As a function of the number of parallel sessions, k, and the round complexity of the protocol, m, the bound is roughly $k^{1/m}$.

We also construct a modular procedure to amplify simulator-query lower bounds (as above), to generic lower bounds in the black-box concurrent zero-knowledge setting. As a demonstration of our techniques, we give a self-contained proof of the $o(\log n/\log\log n)$ lower bound for the round complexity of black-box concurrent zero-knowledge protocols, first shown by Canetti, Kilian, Petrank and Rosen (STOC 2002). Additionally, we give a new lower bound regarding constant-round black-box concurrent zero-knowledge protocols: the running time of the black-box simulator must be at least $n^{\Omega(\log n)}$.

Keywords: Zero-Knowledge, Knowledge Tightness, Concrete Security, Concurrent Zero-Knowledge Lower Bounds.

*Chung is supported by a Simons Foundation Fellowship. Department of Computer Science, Cornell University, Upson Hall 4108, Ithaca, NY 14850, USA. chung@cs.cornell.edu.

†Pass is supported in part by a Alfred P. Sloan Fellowship, Microsoft New Faculty Fellowship, NSF CAREER Award CCF-0746990, AFOSR YIP Award FA9550-10-1-0093, and DARPA and AFRL under contract FA8750-11-2-0211. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US government. Department of Computer Science, Cornell University, Upson Hall, Ithaca, NY 14850, USA. rafael@cs.cornell.edu.

‡Department of Computer Science, Cornell University, Upson Hall, Ithaca, NY 14850, USA. wdtseng@cs.cornell.edu.

1 Introduction

Zero-knowledge interactive proofs, introduced by Goldwasser, Micali and Rackoff [GMR89] are paradoxical constructions allowing one player (called the prover) to convince another player (called the verifier) of the validity of a mathematical statement $x \in L$, while providing no additional knowledge to the verifier. In addition to being an independent construction of interest, zero-knowledge have become an extremely useful tool in construction of numerous cryptographic protocols.

A fundamental question regarding zero-knowledge protocols is whether their composition remains zero-knowledge. In theoretical constructions as well as in practice, a zero-knowledge protocol is sometimes composed in parallel (to amplify soundness or to improve efficiency, for example). It is well-known that the definition of zero-knowledge (ZK) is not closed under parallel composition [GK96b]. Nevertheless, we know numerous constructions of constant-round zero-knowledge protocols that are secure when composed in parallel [FS90, GK96a, Gol02]. As a result, the subject of ZK with respect to parallel composition is widely considered closed.

We turn our attention to another fundamental question regarding zero-knowledge:its knowledge tightness. In its original definition, the zero-knowledge property is formalized by requiring that the view of any probabilistic polynomial time (PPT) verifier V in an interaction with a prover can be "indistinguishably reconstructed" by a PPT simulator S that interacts with no one. Since whatever V "sees" in the interaction can be reconstructed by the simulator, the interaction does not yield any knowledge to V that V can already compute by itself. Because the simulator is allowed to be an arbitrary PPT machine, this traditional notion of ZK only guarantees that the *class* of PPT verifiers learn nothing.

To more concretely measure the knowledge gained by a particular verifier, Goldreich, Micali and Wigderson [GMW91] (see also [Gol01]) put forward the notion of *knowledge tightness*: informally, the "tightness" of a simulation is the ratio of the (expected) running-time of the simulator, divided by the (worst-case) running-time of the verifier. Thus, in a knowledge-tight ZK proof, the verifier is expected to gain no more knowledge than what it could have computed in time closely related to its *worst-case* running-time. In addition to theoretical interests, the knowledge tightness of a zero-knowledge protocol is a helpful aid for setting the security parameter in practice. It is easy to check that the original zero-knowledge protocols [GMR89, GMW91, Blu86] all enjoy constant knowledge tightness. The aforementioned protocols secure under parallel composition [FS90, GK96a, Gol02] also enjoy constant knowledge tightness when executed in isolation; however, when composed in parallel, the tightness of these protocols seem increase/loosen linearly (sometimes even quadratically) with respect to the number of parallel sessions (based on the currently known analysis of their simulators)!

Since we do want to execute zero-knowledge protocols in parallel (for instance in the application of secure multi-party computation), a natural question is to ask: how does the knowledge tightness of a protocol vary when we increase the number of parallel repetitions?

1.1 Our results

In this work we give essentially tight upper and lower bounds to the above question. Our results focus on *black-box* zero-knowledge and "simulator queries", which we explain below.

Informally, a protocol is *black-box* zero-knowledge if there exists a *universal* simulator S, called the *black-box simulator*, such that S generates the view of any adversarial verifier V^* if S is given black-box access to V^*. Essentially all known constructions of zero-knowledge (with the notable exception of [Bar01]) and all practical zero-knowledge protocols are black-box zero-knowledge. Given a black-box simulator S, we focus on bounding the number of black-box queries made by

S to a given adversarial verifier V^*; we refer to this as the *simulator-query* complexity. It is easy to see that the number of queries made by a black-box simulator is closely related to knowledge tightness; in fact, for the case of constant round protocols, they are asymptotically equivalent.

We state our main theorems below:

Theorem 1. *Let n be the security parameter. For any $m = m(n)$, there exists a $2m+7$-round black-box zero-knowledge argument Π for all of NP based on one-way functions, with perfect completeness and negligible soundness error, such that for any polynomially bounded $k = k(n)$, the parallel composition of k-copies of the protocol, Π^k, remains black-box zero-knowledge with simulator-query complexity $O(mk^{1/m} \log^2 n)$.*

The above theorem can be extended to proofs assuming the existence of collision-resistant hash-functions. We complement Theorem 1 with a lower bound:

Theorem 2. *Let n be the security parameter, L be a language, and $m = m(n) \in O\left(\frac{\log n}{\log \log n}\right)$. Suppose Π is a $m(n)$-round black-box zero-knowledge argument for L with perfect completeness and negligible soundness error, and suppose there exist a polynomially bounded $k(n) \geq n$ such that the parallel composition of k-copies of the protocol, Π^k, remains black-box zero-knowledge with simulator-query complexity $O(k^{1/m}/(\log^2 n))$. Then, $L \in \mathsf{BPP}$.*

For protocols with sub-logarithmic number of rounds, Theorem 1 and 2 are tight up to logarithmic factors in the security parameter; essentially, the simulator-query complexity is asymptotically close to $k^{1/m}$ (in most cases, think of k as a low polynomial in n). We mention that one can achieve simulator-query complexity $O(m)$ (independent of k) when $m = \omega(\log n)$.

Briefly, our results show that the concrete security of constant-round black-box zero-knowledge protocols actually decays polynomially in the number of parallel sessions. Fortunately, this decay can be significantly slowed if we consider protocols with more rounds (even if we simply use a large constant m).

1.2 Related Works

While we are unaware of any past work that explicitly studies the knowledge tightness of parallelized zero-knowledge protocols, there are numerous related publications that focus on the composition of zero-knowledge protocols, or on the concrete security of zero-knowledge simulator. Dwork, Naor and Sahai [DNS04] introduces the notion of *concurrent zero-knowledge* protocols; these protocols must stay zero-knowledge even when composed arbitrarily (a strengthening over parallel composition). Micali and Pass [MP06] introduces the notion of *precision*; in a precise zero-knowledge protocol, the running time of the simulator should be closely related to the running time of the adversarial verifier, on a view by view basis[1] (a strengthening over knowledge tightness).

Even with these stronger requirements, Pandey et. al. [PPS+08] is able to construct protocols that are simultaneously precise and (black-box) concurrent zero-knowledge. Note that our results are incomparable with the result of [PPS+08] for many reasons, one of which being that black-box concurrent zero-knowledge protocols require logarithmically many rounds [CKPR01], while our setting is mainly interesting for sub-logarithmic-round protocols. Interestingly, [PPS+08] actually gives a construction of a family of precise concurrent zero-knowledge protocols, with trade-offs between round-complexity and precision, much like our observed trade-off between round-complexity and knowledge tightness for the case of parallelized zero-knowledge.

[1] For example, to achieve precision 2, if the simulator S generates a view of V^* and the running time of V^* on that view is T, then the simulator S must have run in time $2T$.

1.3 Connection to Concurrent Zero-Knowledge

In the second part of our paper, we present a connection from simulator-query lower bounds for zero-knowledge, to round-complexity lower bounds for concurrent zero-knowledge (cZK).

We start by describing the common framework for all known black-box zero-knowledge lower bounds (e.g., [KPR98, Ros00, CKPR01, BL02, Kat08, HRS09, PTW09]). Let Π be a protocol for a language L. To show that Π cannot be zero-knowledge unless the language L is trivial (i.e., $L \in \mathsf{BPP}$), we start by constructing a decision procedure for L. Let S be the black-box zero-knowledge simulator of Π, and let V^* be some "hard to simulate" adversarial verifier, and consider the following decision procedure $\mathcal{D}$: on input x, $\mathcal{D}(x)$ accepts if and only if $S^{V^*}(x)$ generates an accepting view of $V^*(x)$. Usually, the completeness of $\mathcal{D}$ follows easily from the zero-knowledge property; to show that $\mathcal{D}$ is sound often requires more work. Our query-complexity lower bounds (Theorem 2) also follow the same framework. That is, we construct some adversarial verifier V^*_{para} that schedules multiple sessions in parallel, and show that for any zero-knowledge simulator S with appropriately bounded query-complexity, if $x \notin L$, then $S^{V^*_{\text{para}}}(x)$ cannot generate an accepting view of $V^*_{\text{para}}(x)$.

Inspired by the work of Canetti, Kilian, Petrank and Rosen [CKPR01], we next present a modular construction of a concurrent adversarial verifier V^*_{conc} whose purpose is to *amplify* query-complexity lower bounds of more basic verifiers. For example, consider V^*_{para}, an adversarial verifier that is restricted to parallel composition. Our modular construction would take V^*_{para} as input, and output an adversarial verifier $V^*_{\text{conc}} = V^*_{\text{conc}}(V^*_{\text{para}})$ that, among other things, nests multiple incarnations of V^*_{para} in a way that takes full advantage of the concurrent scheduling. Under appropriate parameters, our analysis would conclude that for any zero-knowledge simulator S with *polynomially* bounded query-complexity, if $x \notin L$, then $S^{V^*_{\text{conc}}}(x)$ cannot generate an accepting view of $V^*_{\text{conc}}(x)$ (recall again that this is the key step for most zero-knowledge lower bounds).

To demonstrate our framework, we re-prove the result of [CKPR01] — a $o(\log n/\log\log n)$ round-complexity lower bound for black-box concurrent zero-knowledge (the currently best known round-complexity lower bound); we believe the resulting analysis is quite clean. We also give a second lower bound concerning constant-round cZK protocols:

Theorem (Informal)**.** *Let L be a non-trivial language, and let Π be a constant-round black-box concurrent zero-knowledge protocol with a potentially possibly super-polynomial time simulator. Then the simulator must run in time $n^{\Omega(\log n)}$.*

Incidentally, Pass and Venkitasubramaniam [PV08] do construct constant-round black-box concurrent zero-knowledge protocols for all of NP in the model where both the simulator and the adversarial verifier runs in quasi-polynomial time $n^{\text{poly}(\log n)}$.

We also find our modular framework satisfying on a philosophical level: it serves as an framework in which lower bounds for restricted compositions of zero-knowledge (in this example parallel composition) can be transformed into lower bounds for zero-knowledge in the fully concurrent setting. A similar and celebrated example occurs in the work of Goldreich [Gol02], where it is shown that constructions of zero-knowledge protocols secure under parallel composition directly leads to constructions of concurrent zero-knowledge protocols secure in the timing model.

2 Preliminaries

We use $\mathbb{N}$ to denote the natural numbers $\{0, 1, \ldots\}$, $[n]$ to denote the set $\{1, \ldots, n\}$, and $|x|$ to denote the length of a string $x \in \{0,1\}^*$. By $\mathsf{ngl}(\mathsf{n})$, we mean a function negligible in n (i.e., $1/n^{\omega(1)}$). We assume familiarity with indistinguishability.

2.1 Interactive Protocols

An interactive protocol Π is a pair of interactive Turing machines, (P, V), where V is probabilistic polynomial time (PPT). P is called the prover, while V is called the verifier. $\langle P, V \rangle (x)$ denotes the random variable (over the randomness of P and V) representing V's output at the end of the interaction on common input x. If additionally V receives auxiliary input z, we write $\langle P(x), V(x, z) \rangle$ to denote V's output. We assume WLOG that Π starts with a verifier message and ends with a prover message, and say Π has k **rounds** if the prover and verifier each sends k messages alternately. A full or partial **transcript** of Π is a sequence of alternating verifier and prover messages, $(v_1, p_1, \dots)$, where v denotes verifier messages and p denotes prover messages.

We may compose an interactive proof in parallel. Let $\Pi^k = (P^k, V^k)$ be the **parallel composition** of k copies of Π; that is, each prover and verifier message in Π^k is just concatenation of k independent copies of the corresponding message in Π. Upon completion, V^k accepts if and only if all k sessions are accepted by V. We note that an adversarial verifier may choose to abort in one session but not another.

2.2 Zero Knowledge Protocols

In the setting of zero knowledge, we consider an adversarial verifier that attempts to "gain knowledge" by interacting with an honest prover. An m-session **concurrent adversarial verifier** V^* is a probabilistic polynomial time machine that, on common input x and auxiliary input z, interacts with $m(|x|)$ independent copies of P concurrently (called **sessions**); the traditional stand-alone adversarial verifier is simply a 1-session adversarial verifier. There are no restrictions on how V^* schedules the messages among the different sessions, and V^* may choose to abort some sessions but not others. Let $\text{View}^P_{V^*}(x, z)$ be the random variable that denotes the **view** of V^* in an interaction with P (this includes the random coins of V^* and the messages received by V^*).

A **black-box simulator** S is a probabilistic polynomial time machine that is given black-box access to V^* (written as S^{V^*}). Formally, S fixes the random coins r of V^* a priori, and S is allowed to specify a valid partial transcript $\tau = (v_1, p_1, \dots, p_i)$ of V^*_r, and query V^*_r for the next verifier message v_{i+1}. Here, τ is **valid** if it is consistent with V^*_r, i.e., each verifier message v_j in τ is what V^*_r would have responded given the previous prover messages $p_1, \dots, p_{j-1}$ and the fixed random tape r. Note that S is allowed to "rewind" V^* by querying V^* with different partial transcripts that shares a common prefix.

Intuitively, an interactive proof is zero-knowledge (ZK) if the view of any stand-alone (1-session) adversarial verifier V^* can be generated by a simulator. The protocol is concurrent ZK (cZK) if the view of any concurrent adversarial verifier can be generated as well. The formal definitions follow.

Definition 1 (Black-Box Zero-Knowledge [GMR89, GO94])**.** Let $\Pi = \langle P, V \rangle$ be an interactive proof (or argument) for a language L. Π is **black-box zero-knowledge** if there exists a black-box simulator S such that for every common input x, auxiliary input z and every adversary V^*, $S^{V^*(x,z)}(x)$ runs in time polynomial in $|x|$, and the ensembles $\{\text{View}^P_{V^*}(x, z)\}_{x \in L, z \in \{0,1\}^*}$ and $\{S^{V^*(x,z)}(x)\}_{x \in L, z \in \{0,1\}^*}$ are computationally indistinguishable as a function of $|x|$.

Definition 2 (Black-Box Concurrent Zero-Knowledge [DNS04])**.** Let $\Pi = \langle P, V \rangle$ be an interactive proof (or argument) for a language L. Π is **black-box concurrent zero-knowledge** if for every polynomials m, there exists a black-box simulator S_m such that for every common input x, auxiliary input z and every m-session concurrent adversary V^*, $S_m^{V^*(x,z)}(x)$ runs in time polynomial in $|x|$, and the ensembles $\{\text{View}^P_{V^*}(x, z)\}_{x \in L, z \in \{0,1\}^*}$ and $\{S_m^{V^*(x,z)}(x)\}_{x \in L, z \in \{0,1\}^*}$ are computationally indistinguishable as a function of $|x|$.

2.3 Other Primitives

We informally define the primitives we use to construct zero-knowledge protocols.

Witness-Indistinguishable (WI) Proofs [FS90]: Roughly speaking, an interactive proof is witness indistinguishable if the verifier's view is "independent" of the witness used by the prover for proving the statement. Slightly more formally, a proof is witness-indistinguishable if the views of the verifier in interactions with a prover using two different witnesses are computationally indistinguishable.

Proofs of Knowledge (POK) [FS90, BG02]: 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.

Special-Sound (SS) Proofs [CDS94]: A k-round *public-coin* (i.e., Arthur-Merlin) proof for language $L \in \mathsf{NP}$ with witness relation R_L is special-sound with respect to R_L if the following holds. There exists a deterministic polynomial-time procedure that can extract a witness given two accepting transcript for the protocol, $(\vec{\alpha}, \beta, \gamma)$ and $(\vec{\alpha}', \beta', \gamma')$ satisfying $\vec{\alpha} = \vec{\alpha}'$ but $\beta \neq \beta'$. Here $\vec{\alpha}$ denotes all of the last two messages of the transcript, β denotes the final public-coin verifier challenge, and γ denotes the final prover response.

In our construction of zero-knowledge arguments we use 4-round WI and SS-POK based on one-way functions. This can be instantiated by repeating the Blum Hamiltonicity protocol [Blu86] in parallel, using 2-round statistically binding commitments constructed from one-way functions ([Nao91, HILL99]). We also ask that the length of the final prover challenge be $|\beta| = n^2$, which is possible with sufficient parallel repetition.

3 Construction

We define a zero-knowledge argument ParallelZK in Section 3.1, and show that it satisfies Theorem 1 in Section 3.2.

3.1 The Protocol

Our ZK argument ParallelZK (also used in [PV08, PTV10]) is a slight variant of the precise ZK protocol of [MP06], which in turn is a generalization of the Feige-Shamir protocol [FS89]. The protocol for language $L \in \mathsf{NP}$ proceeds in three stages, given a security parameter n, a common input statement $x \in \{0,1\}^n$, and a round-parameter m:

Stage Init: The verifier picks two random strings $r_1, r_2 \in \{0,1\}^n$ and sends their images $c_1 = f(r_1)$, $c_2 = f(r_2)$ through a one-way function f to the prover. The verifier then acts as the prover in m parallel instances of a 4-round witness indistinguishable and special sound proof of knowledge (WI and SS-POK) of the NP statement "c_1 or c_2 is in the image set of f" (a witness here would be a pre-image of c_1 or c_2). All but the last two messages of each SS-POK is exchanged in this stage; we denote their partial transcripts by $(\vec{\alpha}_1, \vec{\alpha}_2, \ldots, \vec{\alpha}_m)$.

Stage 1: m rounds of message exchanges occur in Stage 1. In the j^{th} round, the prover sends β_j, a random second last message of the j^{th} SS-POK, and the verifier replies with the last message γ_j of the proof. These m rounds are called *slots*. Slot i is *convincing* if the verifier produces an accepting proof (i.e., the transcript $(\vec{\alpha}_i, \beta_i, \gamma_i)$ is accepting). If there is ever an *unconvincing* slot, the prover aborts the whole session.

Stage 2: The prover provides a 4-round witness indistinguishable proof of knowledge (WI-POK) of knowledge of the statement "$x \in L$, or one of c_1 or c_2 is in the image set of f".

Completeness and soundness follows directly from the proof of Feige and Shamir [FS89]; in fact, the protocol is an instantiation of theirs. Intuitively, to cheat in the protocol a prover must "know" an inverse to c_1 or c_2 (because Stage 2 is an argument of knowledge), which requires the prover to invert the one-way function f (it is shown in [FS90] that Stage Init and Stage 1 of the protocol cannot aid the prover in inverting f). A formal description of protocol ParallelZK is shown in Figure 1in appendix.

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 $r_1, r_2 \in \{0,1\}^n$.

V $\rightarrow$ P: $c_1 = f(r_1), c_2 = f(r_2)$.

V $\leftrightarrow$ P: Exchange in parallel (interactively) all but the last two messages $\vec{\alpha}_1, \ldots, \vec{\alpha}_m$ of m WI and SS-POKs on common input (c_1, c_2) with respect to the witness relation:

$$R_{L'}(c_1, c_2) = \{r : f(r) = c_1 \text{ or } c_2\}$$

Note that V acts as the prover in these SS-POK's.

Stage 1: For $j = 1$ to m, exchange the i^{th} "slot"

P $\rightarrow$ V: The second last message β_i of the i^{th} SS-POK.

V $\rightarrow$ P: The last message γ_i of the i^{th} SS-POK.

P aborts if $(\vec{\alpha}_i, \beta_i, \gamma_i)$ is not a valid SS-POK.

Stage 2:

P $\leftrightarrow$ V: a 4-round computational-WI proof of knowledge from P to V on common input (c_1, c_2, x) with respect to the witness relation:

$$R_{L' \vee L}(c_1, c_2, x) = \{(r, w) : r \in R_{L'}(c_1, c_2) \text{ or } w \in R_L(x)\}$$

Figure 1: ParallelZK: a ZK argument for NP with round parameter m.

Remark 1. We note that here we use multiple slots to improve the knowledge tightness of parallel zero knowledge, whereas previously, multiple slots was typically used to achieve concurrent zero knowledge and $\omega(\log n)$ slots were considered. In contrast, we show that in the context of parallel zero knowledge, using even constant number of slots improves the knowledge tightness significantly. Indeed, both our simulation technique and its analysis presented in the next section are new, where we rewind each slot to resolve *all* sessions in parallel (as opposed to previous works that focused on one session at a time).

3.2 The Simulator

To show that protocol Π = ParallelZK satisfies Theorem 1, given any polynomially bounded $k = k(n)$, we need to construct a black-box zero-knowledge simulator $S = S_k$ for protocol Π^k (ParallelZK repeated k times in parallel). On a very high-level, our simulator follows that of Feige and Shamir [FS90]: after fixing the SS-POK prefixes in Stage Init, the simulator rewinds one of the "slots" in Stage 1 (the last two messages of the SS-POKs). If the verifier responds with two convincing slots, the simulator uses the special-soundness property to extract a "fake witness" r such that $f(r) = c_1$ or c_2, and uses this fake witness to simulate Stage 2 of the protocol.

Given an adversarial verifier V^* (for protocol Π^k) and a common input $x \in \{0,1\}^n$, the simulator $S^{V^*}(x)$ does the following:

1. The simulator S interacts with V^*, following the honest prover strategy, until the end of Stage 1. We call this the *reference simulation*.

2. The simulator S attempts to *resolve* all k parallel sessions in the reference simulation by extracting a fake witness r from the SS-POKs for each non-aborting session; aborted sessions are automatically considered resolved (and no fake witnesses are needed). To do so, S repeats the following step (called a rewinding pass) as many times as necessary, until all sessions are resolved.

3. **A rewinding pass.** For each slot i, the simulator rewinds the reference simulation back to the beginning of slot i, sends V^* a fresh random message β_i', and receives a new reply γ_i' (of course this is done in parallel for all k sessions). Note that for each unresolved session j, S already knowns an accepting transcript $(\vec{\alpha}_i, \beta_i, \gamma_i)$ of SS-POK from the reference simulation. If session j does not abort during slot i in this rewinding pass, then S learns another accepting transcript $(\vec{\alpha}_i, \beta_i', \gamma_i')$ of SS-POK. In this case, S can resolve the session j by extracting a fake witness using the special-sound property.

4. S completes the reference simulation using extracted fake witnesses to simulate the Stage 2 proof (only needed in each parallel session that did not abort). S outputs the view of V^* on the reference simulation and this completion.

For simplicity, we assume that for sessions that did not abort in the reference simulation, the extraction of fake witnesses always succeeds whenever S receives an accepting slot in a rewinding pass (i.e., we assume that S never sends the same value for β twice). This assumption can be made without loss of generality by the following modifications of the simulation strategy.

- Let the simulator S performs at most 2^n rewinding passes. If there exist any unsolved sessions j after 2^n rewinding passes, S resolves the session by brute force, i.e., by directly inverting the one-way function f to obtain a fake witness of length n. This modification increases the running time (but not the number of queries) of S by at most a $\text{poly}(n)$ factor (multiplicatively), and makes sure that S makes at most $\text{poly}(2^n)$ queries to V^*.

- Let the final verifier challenge in the SS-POK have length $|\beta| = n^2$. In this case, the probability of S ever querying V^* with the same value of β twice is $\text{poly}(2^n) \cdot 2^{-n^2} = 2^{-\Omega(n^2)}$, definitely negligible in n.

We now show two lemmas regarding S that together show that ParallelZK is zero-knowledge when composed in parallel.

Lemma 3. *S runs in expected polynomial time, and makes $O(mk^{1/m} \log^2 n)$ queries in expectation.*

Lemma 4. *On common input $x \in L$, the output of S is indistinguishable from the real view of V^*.*

We defer the proof of Lemma 3 to the next section, where we bound the *expected number of rewinding passes* before S extracts all necessary fake witnesses. We give a sketch of proof of Lemma 4 now.

Proof Sketch of Lemma 4. The output of S up to the end of Stage 1 (i.e., the reference simulation) is identical to the view of V^*, because S follows the honest prover strategy. The output of S in Stage 2 of the protocol is computationally indistinguishable from the view of V^* because the Stage 2 proof is witness indistinguishable. Formally, this can be shown with a hybrid argument where we incrementally exchange each of the k parallel Stage 2 proofs from using "fake witnesses" r such that $f(r) = c_1$ or c_2 (the simulator strategy), to a real witnesses w for $x \in L$ (the honest prover strategy). □

3.3 Proof of Lemma 3

In this section, we prove Lemma 3 by bounding the expected number of rewinding passes in an execution of S. Let R be a random variable that denotes the number of rewinding passes. We will show that:

$$\mathbb{E}[R] = \mathbb{E}[\#\text{ rewinding passes }] \leq O(k^{1/m} \cdot \log^2 n).$$

This then implies Lemma 3 because outside of rewinding passes, $S^{V^*}(x)$ makes only $O(m)$ queries to V^* and runs in polynomial time.

Before presenting our analysis for the general case of m slots, we revisit the classical analysis for the case of single slot for intuition.

The case of single slot. The analysis is very simple. For every $j \in [k]$, let R_j denote the number of rewinding passes to resolve session j, and let p_j be the probability that session j does not abort during the single slot. Recall that session j is resolved if it aborts in the reference simulation, and otherwise, the simulator needs to rewind the slot several times until session j does not abort again. Hence, the expected number of rewinding passes to resolve session j is

$$\mathbb{E}[R_j] = (1 - p_j) \cdot 0 + p_j \cdot \frac{1}{p_j} = 1.$$

By linearity of expectation, the expected number of rewinding passes is

$$\mathbb{E}[R] = \sum_j \mathbb{E}[R_j] = k \leq O(k \cdot \log^2 n).$$

We note that the above simple analysis is tight. Consider the case where during the slot, each session aborts independently with probability $(1 - 1/k)$. It is not hard to see that in this case, with constant probability, at least one session does not abort during the slot, and the simulator needs to rewind k times in expectation to resolve the survival session. Therefore, the expected number of rewinding passes is $\Omega(k)$.

In fact, it is instructive to note that the following natural generalization of the above example is essentially the worse-case example for the general case of m slots: during each slot $i \in [m]$, each survival session j aborts independently with probability $(1 - k^{-1/m})$. In this case, each session does not abort during the m slots with probability $(k^{-1/m})^m = 1/k$, and hence with constant probability, at least one session survives after m slots. Resolving the survival session requires $k^{1/m}/m$ rewinding passes in expectation, and hence the expected number of rewinding passes is $\Omega(k^{1/m}/m)$.

We note that although in the above example, each session aborts during each slot independently, in general, the aborting probability of each session at each slot can depends arbitrarily on the history and correlated arbitrarily.

The general case of m slots. To analyze the expected number of rewinding passes, we define the following $[0,1]$-valued random variables based on the reference simulation generated in Step 1. Let h_i denote the partial transcript of the reference simulation before slot i. For every slot $i \in [m]$ and session $j \in [k]$, we define random variable $p_{i,j}$ as follows.

- If session j is already aborted at the end of slot i, then we define $p_{i,j} \triangleq 1$.
- Otherwise, we define $p_{i,j}$ to be the conditional probability

$$p_{i,j} \triangleq \Pr[\text{ session } j \text{ does not abort during slot } i \mid h_i].$$

For intuition, $p_{i,j}$ is essentially the probability that S can resolve session j by rewinding slot i. Now consider the *best slot* for each session — the slot with the highest $p_{i,j}$ value (this is the slot that S wants to rewind). We record this value as

$$p_j^* = \max_i p_{i,j}$$

Note that for a session j that aborts in the reference simulation, we have $p_j^* = 1$, indicating that sessions j is already resolved and matching the above intuition. Finally, the number of rewinding passes depends heavily on the *worst session* — the session with the worst p_j^* value (the "worst best slot"). We record this value as the *critical probability*:

$$p^* = \min_j p_j^*.$$

To see how the critical probability p^* plays an important role in the expected number of rewinding passes, note that on one hand, S needs roughly $1/p^*$ rewinding passes to resolve the worse-case session; on the other hand, the chance of having a reference simulation with small critical probability (say, $p^* \leq p$) is rare (at most p^m). Therefore, to upper bound $\mathbb{E}[R]$, we define the following events, which partition the probability space according to the critical probability. For every $t \in \mathbb{N}$, let

$$\alpha_t \stackrel{\text{def}}{=} \left(\frac{1}{2^t \cdot k^{1/m}}\right)$$

- Let A_0 be the event that $p^* \geq \alpha_0 = k^{-1/m}$, and for every $t \in \mathbb{N}$, let A_t be the event that

$$\alpha_t \leq p_j^* < \alpha_{t-1}.$$

Similarly for every session $j \in [k]$,

- Let $A_{0,j}$ be the event that $p_j^* \geq \alpha_0 = k^{-1/m}$, and for every $t \in \mathbb{N}$, let $A_{t,j}$ be the event that

$$\alpha_t \leq p_j^* < \alpha_{t-1}.$$

We can now express the expectation of the number of rewinding passes as follows.

$$\begin{aligned}\mathbb{E}[R] &= \sum_{t\geq 0} \Pr[A_t] \cdot \mathbb{E}[R \mid A_t] \\ &\leq \Pr[A_0] \cdot \mathbb{E}[R \mid A_0] + \sum_{t\geq 1} \left(\sum_{j=1}^{k} \Pr[A_{t,j}] \right) \cdot \mathbb{E}[R \mid A_t],\end{aligned}$$

where the last inequality follows by $A_t \subseteq \cup_j A_{t,j}$ (which follows from definition). We proceed to bound each term. For A_0, we use trivial bound $\Pr[A_0] \leq 1$. For general $t \geq 1$ and every $j \in [k]$, we first observe that when $A_{t,j}$ happens, session j does not abort all of its m slots in the reference simulation (since otherwise, $p_j^* = 1$). This happened despite the fact that each slot i in session j in the reference simulation could have only survived (not aborted) with probability $p_{i,j} \leq \alpha_{t-1}$. Thus,

$$\Pr[A_{t,j}] \leq \alpha_{t-1}^m = \left(\frac{1}{2^{t-1} \cdot k^{1/m}} \right)^m = \frac{1}{2^{m(t-1)} \cdot k},$$

and,

$$\sum_{j=1}^{k} \Pr[A_{t,j}] \leq k \cdot \frac{1}{2^{m(t-1)} \cdot k} = \frac{1}{2^{m(t-1)}}.$$

It remains to bound $\mathbb{E}[R \mid A_t]$, which is given in the follow lemma.

Lemma 5. *For every $t \geq 0$, we have*

$$\mathbb{E}[R \mid A_t] \leq O\left(2^t \cdot k^{1/m} \cdot \log^2 n\right).$$

We apply Lemma 5 to upper bound $\mathbb{E}[R]$ first.

$$\begin{aligned}\mathbb{E}[R] &\leq \mathbb{E}[R \mid A_0] + \sum_{t\geq 1} \frac{1}{2^{m(t-1)}} \cdot \mathbb{E}[R \mid A_t] \\ &\leq O\left(k^{1/m} \cdot \log^2 n\right) + \sum_{t\geq 1} \frac{2^t}{2^{m(t-1)}} \cdot O\left(k^{1/m} \cdot \log^2 n\right) \\ &\leq O\left(k^{1/m} \cdot \log^2 n\right).\end{aligned}$$

This completes the proof of Lemma 3.

Proof of Lemma 5. The event A_t means that in the reference simulation, for every non-aborting session j, there exists a useful slot $i \in [m]$ such that

$$\Pr[\text{ session } j \text{ is not aborted after slot } i \mid h_i] = p_{i,j} \geq \alpha_t.$$

Therefore, in each rewinding pass, the simulator S may learn an (additional) accepting transcript of SS-POK in session j with probability at least α_t, allowing it to extract a fake witness.

Fix a non-aborting session j, and define

$$q = \left(\frac{10 \cdot \log^2 n}{\alpha_t} \right) = O\left(2^t \cdot k^{1/m} \cdot \log^2 n\right),$$

Because the rewinding passes are independent, we have

$$\Pr[\text{session } j \text{ is resolved after } q \text{ rewinding passes}] = 1 - (1-\alpha_t)^q \geq 1 - \mathsf{ngl}(\mathsf{n}).$$

Since there are at most k survival sessions, by the union bound,

$$\Pr[\text{all sessions are resolved after } q \text{ rewinding passes}] \geq 1 - \mathsf{ngl}(\mathsf{n}).$$

In other words, every q rewinding passes can solve all the sessions with probability at least $1-\mathsf{ngl}(\mathsf{n})$. It follows that

$$\begin{aligned}\mathbb{E}[R \mid A_t] &\leq (1-\mathsf{ngl}(\mathsf{n}))\cdot q + \mathsf{ngl}(\mathsf{n})\,(1-\mathsf{ngl}(\mathsf{n}))\cdot 2q + \mathsf{ngl}(\mathsf{n})^2\,(1-\mathsf{ngl}(\mathsf{n}))\cdot 3q + \cdots \\ &\leq O(q) = O\left(2^t \cdot k^{1/m} \cdot \log^2 n\right). \end{aligned}$$

□

4 Lower Bound

The proof of Theorem 2 follows a well-known framework (e.g., [GK96b, CKPR01]). Let S be a black-box zero-knowledge simulator for $\Pi^k = (P^k, V^k)$ that makes less than $q = O(k^{1/m}/\log^2 n)$ queries, and let V^{k*} be a particular adversarial verifier to be specified later. We define $\mathcal{D}$, a BPP decision procedure for L by combining S and V^{k*}: on input instance x, $\mathcal{D}(x)$ accepts if and only if $S^{V^{k*}}(x)$ outputs an accepting view of V^{k*} (i.e., all k sessions of V^{k*} accept). Using the zero-knowledge property, it is easy to show (see for example [GK96b]) that if the modified protocol $\Pi^{k*} = (P^k, V^{k*})$ is complete for L (based on our choice of V^{k*}), then $\mathcal{D}$ is complete for L as well. The main effort of the proof is to show that $\mathcal{D}$ is sound; this relies both on the choice of V^{k*} and the fact that S makes less than q queries to V^{k*}. We discuss our choice of V^{k*} in Section 4.1, and analyze the soundness of $\mathcal{D}$ in Section 4.2.

4.1 The Random Termination Verifier V^{k*}

In this section, we define a verifier V^{k*} for the parallelized protocol with two goals in mind: the protocol $\Pi^{k*} = (P^k, V^{k*})$ should be complete (so that $\mathcal{D}$ is complete), and V^{k*} should be sound against any rewinding simulator S that makes less than q queries to V^{k*} (so that $\mathcal{D}$ is sound).

Just as [CKPR01], we define V^{k*} to follow the honest verifier strategy V^k with one extra property: random termination.[2] Whenever the prover P^k or the rewinding simulator S makes a query to V^{k*}, V^{k*} determines, with independent and fresh randomness,[3] whether or not to terminate immediately and *accept* with probability $\rho \in [0,1]$, a parameter to be specified later; this is done *independently* for each of the k parallel sessions (i.e., one session may be terminated while other sessions continue). Due to this independence among parallel sessions, we often treat V^{k*} as k machines, $(V_1^*, \ldots, V_k^*)$, each responsible for making the decision to terminate and generating the verifier messages for one session. Note that the fresh randomness is only used to decide whether to terminate or not; V^{k*} generates protocol messages using its default random tape that is kept the same between rewinds (as expected by following the honest verifier strategy).

[2]The term "random termination" was first used by Haitner [Hai09], but the random termination verifier we considered already appeared in the earlier work of [CKPR01].

[3]We use a well-known technique (see for example [GK96b, CKPR01]) to generate fresh independent randomness on the fly for each query from the simulator S, despite the fact that S may rewind V^{k*} between queries and force V^{k*} to use the same random tape. Let $\mathcal{H}$ be a family of q-wise independent hash-functions, and let V^{k*} sample one hash-function $h \leftarrow \mathcal{H}$ in the very beginning. Then whenever V^{k*} receives a query (from P^k or S), V^{k*} applies h to the current protocol transcript (the sequence of messages exchanged in the protocol so far) and use the output as a fresh random tape. Since S makes at most q queries to V^{k*}, the output distribution of the hash-function is truly uniformly random.

Clearly, $\Pi^{k*} = (P^k, V^{k*})$ is still complete. It remains to show that V^{k*} is "sound" against the rewinding S; that is, on input $x \notin L$, $S^{V^{k*}}$ is unlikely to generate an accepting transcript of V^{k*}. From now on we drop the common input $x \notin L$. Intuitively, by randomly terminating, V^{k*} can better protect its randomness against S's rewinds (when V^{k*} terminates, S learns nothing about V^{k*}'s fixed random tape), thus ensuring soundness. To make this intuition more concrete, suppose for example that S made q queries $\tau_1, \ldots, \tau_q$ to V^{k*}, and without loss of generality outputs the view of V^{k*} on a subset of size m of those queries[4], $T = \{\tau_{i_1}, \ldots, \tau_{i_m}\}$. Further suppose that there exists a parallel session $j \in [k]$ such that V^{k*} does not terminate on the queries in T, but terminates on all remaining queries. Then intuitively, S's rewinding does not help S convince V^{k*} in session j, and the soundness of the original protocol Π should imply that V^{k*} rejects with overwhelming probability in session j (and therefore rejects overall).

The core of our proof is to show that, with high probability, for every subset of size m of queries $T = \{\tau_{i_1}, \ldots, \tau_{i_m}\}$ made by S, there exists a session $j \in [k]$ with *overwhelming probability* such that rewinds are "not helpful" for session j with respect to T in the above manner. We make this possible by setting the termination probability to $\rho = (1 - 1/q)$.

We now state the formal lemmas. Let n be the security parameter and L be a language. Suppose there exists a $m(n) \in O\left(\frac{\log n}{\log \log n}\right)$-round argument $\Pi = (P, V)$ for L with perfect completeness and negligible soundness error. For any polynomially bounded $k(n) \geq n$, let S be a black-box zero-knowledge simulator of the parallelized protocol $\Pi^k = (P^k, V^k)$ that makes at most

$$q = k^{1/m}/(\log^2 n)$$

queries, and let V^{k*} be a random termination verifier of the parallelized protocol with termination probability

$$\rho = \left(1 - \frac{1}{q}\right) = \left(1 - \frac{1}{k^{1/m}} \cdot (\log^2 n)\right).$$

(These parameters passes the following sanity checks: q is polynomially bounded and $q \geq m$ — the simulator queries V^{k*} at least once for each round of the protocol. It is also useful later to know that $\binom{q}{m} \leq q^m \leq k$.) Then:

Lemma 6. *On input $x \in L$, $\mathcal{D}(x)$ accepts with probability 1, i.e., $S^{V^{k*}}(x)$ outputs an accepting view of V^{k*} with probability $1 - \mathsf{ngl}(\mathsf{n})$.*

Lemma 7. *On input $x \notin L$, the probability that $S^{V^{k*}}(x)$ generates an accepting view of V^{k*} is negligible, i.e., $\mathcal{D}$ has negligible soundness error.*

We sketch the proof of Lemma 6 now, and give the proof of Lemma 7 in the next section.

Proof Sketch. Using the zero-knowledge property, the output of S is indistinguishable from the view of V^{k*} in an execution with P^k. Therefore it is enough to show that $\langle P^k, V^{k*} \rangle (x)$ accepts with probability 1. In each parallel session $j \in [k]$, V_j^* accepts by definition if it decides to terminate in some protocol round. Otherwise, V_{j^*} is identical to V and would still accept with probability 1 because the original protocol $\Pi = (P, V)$ has perfect completeness. □

[4]Without loss of generality, we may assume that before S outputs a view of V^{k*}, S first queries V^{k*} with the messages in the view (if S hasn't already). This may increase the number of queries by m, and thus weaken the resulting lower bound from q to $q - m$. Nevertheless, this does not change our lower bound since $q = \omega(m)$ in Theorem 2.

4.2 Soundness of $\mathcal{D}$

Proof of Lemma 7. We prove Lemma 7 with a reduction. Suppose for the sake of contradiction that S convinces V^{k*} on some input $x \notin L$ with probability more than $1/p(n)$ for some polynomial p. Using S, we construct a cheating prover P^* for the original protocol $\Pi = (P, V)$ that convinces V with non-negligible probability.

Before we start, assume without loss of generality that S makes exactly q queries, and that before S outputs a view of V^{k*}, S would first query V^{k*} on all previous messages in the view. For technical convenience, we let V^{k*} make a fresh decision to terminate for each query and each session, *even if V^{k*} has already terminated previously in the same session.* I.e., regardless of history or message content, for each query and each parallel session, V^{k*} always terminates independently with probability ρ.

Our P^* is a natural extension of the classic reduction of [GK96b] — P^* guesses a session $j_0 \in [k]$ and m indices $T_0 = \{i_1, \ldots, i_m\} \subseteq [q]$ uniformly at random, and interacts with an outside honest V by internally simulating an interaction of (S, V^{k*}) with V embedded in session j_0, queries $\tau_{i_1}, \ldots, \tau_{i_m}$ of V^{k*}. In comparison, the idea of guessing a random query subset is exactly as in [GK96b]. The difference is that the reduction in [GK96b] is for single session protocols, and in contrast, we reduce from parallel protocols to single session protocols. Hence, our reduction P^* guesses a random session as well.

In more details, P^* runs S and V^{k*} internally. It simulates $k-1$ sessions of V^{k*} honestly (except $V^*_{j_0}$). When simulating $V^*_{j_0}$ for the i^{th} query where S queries τ_i, P^* first simulates (with fresh randomness) $V^*_{j_0}$'s decision on termination. If $V^*_{j_0}$ decides to terminate but $i \in T_0$ or if $V^*_{j_0}$ does not terminate but $i \notin T_0$, P^* aborts (in both these cases, the termination decision of $V^*_{j_0}$ is incompatible with P^*'s choice of queries to forward). If the forwarded queries (index set T_0) are not "consistent" (e.g., if they query for the same round of the protocol more than once, or the query contains inconsistent transcript), P^* aborts as well. Note that if P^* does not abort, then V^{k*} is perfectly simulated (even in session j_0).

Now consider the following best case scenario. Suppose that at the end of the simulation, S successfully outputs an accepting view of V^{k*}. Moreover, suppose that the accepting view consists exactly of the queries in index set T_0 (this automatically guarantees that the forwarded queries are consistent), and suppose that P^* does not abort (i.e., termination decisions are compatible with the forwarded queries). Then, P^* will have successfully convinced the outside honest V. The rest of the proof is devoted to show that this best case scenario occurs with noticeable probability (roughly $1/(p \cdot k^2)$).

Let $T \subset [q]$ denote an index set $\{i_1, \ldots, i_m\}$ of size m. For an index set $T \subset [q]$ and a session $j \in [k]$, we define $A(T, j)$ to be the event that, on session j, V^{k*} terminates session j on query τ_i iff $i \notin T$. Referring back to our intuition earlier, $A(T, j)$ denotes the event that for session j, S's rewinds are not helpful with respect to the queries indexed by T. If event $A(T, j)$ holds, and S uses the queries indexed by T to form an accepting view of V^{k*}, and P^* guesses both $T_0 = T$ and $j_0 = j$ in the beginning, then P^* will have successfully convinced the outside honest V.

We claim that by the setting of parameters, we have

$$\Pr[\forall T \subset [q], \exists j \in [k] \text{ s.t. } A(T, j)] \geq 1 - \mathsf{ngl}(\mathsf{n}) \tag{1}$$

where $\mathsf{ngl}(\mathsf{n})$ denotes a negligible quantity in n. In words, with overwhelming probability, for every possible index set T of size m that S may use to output a view of V^{k*}, there exists a session j such that S's rewinds are not helpful with respect to the queries indexed by T.

Before proving (1), we first use the claim to show that P^* convinces V with noticeable probability. Recall that S outputs an accepting view of V^{k*} with probability $1/p$. By a union bound, we

have

$$\Pr[(S \text{ outputs accepting view of } V^{k*}) \wedge (\forall T \subset [q], \exists j \in [k] \text{ s.t. } A(T,j))] \geq (1/p) - \mathsf{ngl}(\mathsf{n}).$$

Note that when the above event holds, there exist a unique index $\hat{T}$ of m queries used by S to form an accepting view of V^{k*}, and there exists a session $\hat{j} \in [k]$ such that $A(\hat{T}, \hat{j})$ holds. As mentioned earlier, if P^* guesses $j_0 = \hat{j}$ and $T_0 = \hat{T}$ correctly, P^* will have successfully convinced V. Since P^* guesses j and T uniformly at random and independent of the interaction between S and V^{k*}, we have

$$\begin{aligned}
&\Pr[P^* \text{ convinces } V] \\
&\geq \Pr[(S \text{ convinces } V^{k*}) \wedge (\forall T \subset [q], \exists j \in [k] \text{ s.t. } A(T,j)) \wedge (P^* \text{ guesses } \hat{T} \text{ and } \hat{j} \text{ correctly})] \\
&\geq \frac{(1/p - \mathsf{ngl}(\mathsf{n}))}{k \cdot \binom{q}{m}} \geq \frac{1}{p \cdot k^2},
\end{aligned}$$

where in the last line we used $\binom{q}{m} \leq q^m \leq k$. This contradicts to the fact that Π has negligible soundness error and completes our analysis.

It remains to show (1). By definition, each session j terminates on each query τ_i with probability exactly ρ, independent from any other session or query. Hence, for any session j and index set T of size m, the probability that event $A(T,j)$ holds is

$$\Pr[A(T,j)] = \rho^{q-m} \cdot (1-\rho)^m \geq \left(1 - \frac{1}{q}\right)^q \cdot \left(\frac{1}{q}\right)^m \geq \Omega\left(\frac{1}{k} \cdot (\log^{2m} n)\right).$$

It follows that

$$\Pr[\exists j \in [k] \text{ s.t. } A(T,j)] \geq 1 - \left(1 - \Omega\left(\frac{1}{k} \cdot (\log^{2m} n)\right)\right)^k \geq 1 - e^{-\Omega(\log^{2m} n)}.$$

Finally, by a union bound, we have

$$\Pr[\forall T \subset [q], \exists j \in [k] \text{ s.t. } A(T,j)] \geq 1 - e^{-\Omega(\log^{2m} n)} \cdot \binom{q}{m} \geq 1 - \mathsf{ngl}(\mathsf{n}),$$

as claimed. □

As with most lower bounds for black-box zero-knowledge, a careful reading reveals that Theorem 2 also applies to more liberal definitions of zero-knowledge, such as ε-zero-knowledge and zero-knowledge with expected polynomial time simulators. Additionally, note that the proof of Lemma 7 never assume that S is a zero-knowledge simulator, and works just as well for any PPT oracle machine S. We restate Lemma 7 below with this strengthening, which will be useful for deriving round-complexity lower bounds for black-box concurrent zero-knowledge protocols later.

Lemma 7 (Generalized)**.** *Let S be any oracle* PPT *machine that makes at most $q = k^{1/m}/(\log^2 n)$ queries to the oracle. Then for every $x \notin L$, the probability that $S^{V^{k*}}(x)$ outputs an accepting view of V^{k*} is negligible.*

Remark 2. By examining the technical inner workings of the proof of Canetti, Kilian, Petrank and Rosen [CKPR01] (which also uses a random termination verifier), we discovered that part of their analysis implicitly presents a lower bound for the number of queries made by black-box simulators for parallel zero-knowledge protocols. Compared with Theorem 2 and our analysis, the result of

[CKPR01] establishes a weaker bound (and is arguably more complicated); this is not surprising, since establishing a parallel lower bound was not their goal.

Specifically, [CKPR01] implicitly establishes a $\log^{\omega(1)}(k)$ lower bound on the number of simulator queries, whereas we were able to establish a lower bound of $k^{1/m}/(\log^2 n)$. Nevertheless, we believe that by adapting our parameters (which may seem strange for their setting), their analysis could be strengthened to match our lower bounds (we have not verified all the details, however).

5 Round Complexity Lower bound of Concurrent Zero-Knowledge

We re-derive the result of Canetti, Kilian, Petrank and Rosen [CKPR01] — a $\tilde{\Omega}(\log n)$ lower bound on the round complexity of black-box concurrent zero-knowledge (cZK) protocols. More importantly, we do so while *modularizing* the proof of [CKPR01]. We believe that the this modular view not only simplifies the presentation of [CKPR01], but in fact provides new insights regarding the analysis of [CKPR01]. Indeed, as a concrete example, we use our framework to give lower bounds on the simulator running time of constant-round black-box cZK protocols at the end of this section.

Formally, we show that:

Theorem 8 ([CKPR01])**.** *Let n be the security parameter, and let $\Pi = (P, V)$ be a $m(n)$-round concurrent black-box zero-knowledge argument for a language L with perfect completeness and negligible soundness. If $m(n) \in o(\log n / \log\log n)$, then $L \in \mathsf{BPP}$.*

We recall again the general framework as in [GK96b, CKPR01] for proving such lower bounds, and then present a high-level discussion on our modularized approach. Let S be the *concurrent* zero-knowledge simulator of Π. We define a particular concurrent adversarial verifier $V^* = V^*_{\text{conc}}$, and show that the following decision procedure $\mathcal{D}$ decides L in probabilistic polynomial time: $\mathcal{D}(x)$ accepts if and only if $S^{V^*_{\text{conc}}}(x)$ outputs an accepting view of V^*_{conc}.

On a high-level, we construct the concurrent adversarial verifier V^*_{conc} in a modular way – our V^*_{conc} takes as input any "basic verifier" V^*_{base}, and recursively invokes V^*_{base} in the same nested way as [CKPR01]. In other words, all V^*_{conc} does is to schedule many incarnations of a given basic verifier V^*_{base} in a nested way, oblivious to the details of V^*_{base}. To prove Theorem 8, we will instantiate V^*_{base} with the parallel random termination verifier V^{k*} constructed in 4.1. The intuition behinds our construction is as follows.

- We view both parallel repetition and random terminations together as a technique to achieve some simulator-query lower bound for a *more restricted adversarial verifier* V^*_{base} (in this case, restricted to parallel composition).
- We view the recursive schedule as a technique that takes advantage of full concurrency to *amplify* the above simulator-query lower bound up to a super-polynomial quantity, therefore achieving the desired impossibility result / lower bound for cZK.

We note that the resulting concurrent adversarial verifier V^*_{conc} we obtained (when V^*_{base} is instantiated with V^{k*}) is essentially the same as the one constructed in [CKPR01]. However, with our modular analysis, we gain a more refined intuition for the features of V^*_{conc}. This also allows us to establish a quasi-polynomial lower bound on the simulator running time of constant-round black-box cZK without much additional effort.

5.1 Defining the Adversarial Verifier V^*_{conc}

V^*_{conc} recursively invoke the random termination verifier defined in Section 4.1 according to the schedule described in [CKPR01]. As part of our efforts towards a modular analysis, we define V^*_{conc} to take as input any "basic verifier" V^*_{base}; V^*_{conc} then proceed recursively invoke V^*_{base} according to the schedule of [CKPR01]. For the purpose of showing a round-complexity lower bound for black-box cZK, we will instantiate V^*_{base} with the random termination verifier V^{k*} further down in the analysis.

Let V^*_{base} be a concurrent verifier for protocol Π that schedules some polynomial sessions of Π into $m(n)$ rounds of interaction. Let $V^*_{\text{base}}(n, r)$ represent an **incarnation** of V^*_{base}, given security parameter n and random tape r. On input n (a security parameter), d (a nesting depth parameter) and r (a random tape), $V^*_{\text{conc}}(n, d)$ does the following:

Specifying the recursive schedule. Simply put, each call to $V^*_{\text{conc}}(n, d)$ corresponds with an incarnation $V^*_{\text{base}}(n, r)$ for a uniform r, with the additional modification that between every prover query and verifier response, $V^*_{\text{conc}}(n, d)$ nests a recursive call of itself, $V^*_{\text{conc}}(n, d + 1)$, with increased depth; since protocol Π start with a verifier message and end with a prover message, there will be $m - 1$ nested calls.

Formally, V^*_{conc} starts by generating a "fresh random tape" r for V^*_{base} (we clarify this later) and invokes $V^*_{\text{base}}(n, r)$ with the following modification. After every prover query τ for $V^*_{\text{base}}(n, r)$ that expects a verifier response, V^*_{conc} delays the response from $V^*_{\text{base}}(n, r)$, and instead recursively calls in itself with parameters $V^*_{\text{conc-child}} = V^*_{\text{conc}}(n, d + 1)$ (increased depth). We call τ the **initiating query** for $V^*_{\text{conc-child}}$. (Then $V^*_{\text{conc-child}}$ invokes an incarnation $V^*_{\text{base}}(n, r')$ and returns the first verifier message of $V^*_{\text{base}}(n, r')$.)

When $V^*_{\text{conc-child}}$ terminates, signaled by some the final prover query τ' meant for $V^*_{\text{conc-child}}$ (we call τ' the **closing query** for $V^*_{\text{conc-child}}$), V^*_{conc} makes sure that the prover has successfully "convinced" $V^*_{\text{conc-child}}$ (as we will define later, this means that the prover has successfully convinced the incarnation of V^*_{base} invoked by $V^*_{\text{conc-child}}$). At this point, V^*_{conc} generates a verifier response by forwarding the original initiating query τ to $V^*_{\text{base}}(n, r)$. Note that while this is V^*_{conc}'s response to the closing query τ', the response depends only on the randomness r and the initiating query τ (because it is in fact $V^*_{\text{base}}(n, r)$'s response to τ).

When V^*_{conc} receives the closing query for $V^*_{\text{base}}(n, r)$, it accepts if and only if the prover has successfully convinced $V^*_{\text{base}}(n, r)$.

Specifying the recursion depth. When $d = d_{\max} \triangleq \log_m n$, we stop all recursive calls (i.e., $V^*_{\text{conc}}(n, d_{\max}, r)$ answers each prover query without delay).

To help understand the definition, we note that for an incarnation V^*_{base}, the first $m - 1$ round prover message queries are initiation queries, and the last round prover message queries are closing queries. Also, all except for the first round verifier messages of V^*_{base} are responses to initiating queries.

It remains to define how each recursive call of V^*_{conc} generates the randomness r for its own incarnation of V^*_{base}. Just as before, to ensure that we have fresh randomness even when rewound by the simulator, V^*_{conc} generates r using the hash-function trick. Let g be drawn from a family of highly independent[5] hash-functions. In each recursive call of V^*_{conc}, corresponding to an initiating query τ, V^*_{conc} generates r by apply g to τ (note that the query τ is just the current global transcript

[5] If the simulator has running time $T(n)$, we would need a family of $T(n)$-wise independent hash-functions.

of interaction right when V^*_{conc} is about to spawn V^*_{base}).[6] From now on we include that hash-function g as an input to V^*_{conc}.

Given the security parameter n, $V^*_{\text{conc}}(n, g)$ simply starts by running $V^*_{\text{conc}}(n, g, d = 1)$.

5.2 Properties of V^*_{conc}

Observe that each call of $V^*_{\text{conc}}(n)$ makes $m - 1$ recursive calls to itself, and that the recursion terminates at depth $\log_m n$. Therefore, V^*_{conc} calls itself $\approx n$ times in total, spawning $\approx n$ incarnations of V^*_{base}. Each incarnation of $V^*_{\text{base}}(n, r)$ corresponds to a call of $V^*_{\text{conc}}(n, g, d)$ for some depth d, and therefore every incarnation of V^*_{base} can be assigned a depth (although this depth is unbeknown to V^*_{base} itself). In general, we treat each incarnation V^*_{base} as having its own local view, oblivious to the nesting schedule; in contrast, V^*_{conc} has a global view of the whole interaction.

In order to motivate the definition of V^*_{conc}, recall that our main goal/difficulty is to show that $\mathcal{D}$ is sound; that is, on input $x \notin L$, $S^{V^*_{\text{conc}}}(x)$ cannot output an accepting view of V^*_{conc}. Suppose we have a "soundness lower bound" of the form that for all PPT oracle machines $\mathcal{M}$, $\mathcal{M}^{V^*_{\text{base}}}$ cannot generate an accepting view of V^*_{base} unless it makes some minimum required number of queries to V^*_{base}, say q_0 (e.g., the generalized Lemma 7 is one such lemma for the random termination verifier V^{k*}). The high level purpose of V^*_{conc} is to *amplifying* this query requirement, i.e., require that for all PPT oracle machines $\mathcal{M}$, $\mathcal{M}^{V^*_{\text{conc}}}$ cannot generate an accepting view of V^*_{conc} unless it makes some large required number ($\gg q_0$) of queries to V^*_{conc}.

The following property is crucial for this amplification: whenever S makes an initiating query τ to an incarnation of V^*_{base} at depth d, (corresponding to a recursive call of $V^*_{\text{conc}}(n, g, d)$), a nested call of $V^*_{\text{conc-child}} = V^*_{\text{conc}}(n, g, d+1)$ is made; if S wish to see the response of V^*_{base} to τ, S must first generate an accepting transcript of $V^*_{\text{conc-child}}$ (encoded in the closing query corresponding to τ); in particular, it contains an accepting transcript of another incarnation of V^*_{base} at depth $d+1$ spawned by $V^*_{\text{conc-child}}$. Intuitively, this causes a multiplicative blow-up in number of required queries, for simulating an accepting transcript of V^*_{conc}, per each increase in depth of nesting. We formalize this intuition in the following amplification lemma.

For an incarnation V^*_{base} invoked by $V^*_{\text{conc}}(n, g, d)$ in the interaction of S and V^*_{conc}, we say that V^*_{base} is an accepting incarnation, if S ever makes $V^*_{\text{conc}}(n, g, d)$ accept (when S sends a closing query to $V^*_{\text{conc}}(n, g, d)$).

Lemma 9. *Let $q(n) \geq m$ be a parameter. If $S^{V^*_{conc}}$ outputs an accepting view of V^*_{conc}, and during the execution of $S^{V^*_{conc}}$, all accepting incarnations V^*_{base} respond to at least q distinct initiating queries, then S makes at least $q^{d_{\max}}$ queries to V^*_{conc}.*

Proof. We prove the lemma by a (backward) induction on the nesting depth d. In the induction, instead of bounding the number of queries that S makes, we consider the number of responses that V^*_{conc} actually responds to S, which makes our induction cleaner. We say that a query τ is *inside* $V^*_{\text{conc}}(n, g, d)$ if τ is a query to some $V^*_{\text{conc}}(n, g, d')$ (with $d' \geq d$) that is recursively called from $V^*_{\text{conc}}(n, g, d)$.[7] We use the following induction hypothesis.

[6] We fix all the randomness of V^*_{base} at the very beginning of its invocation by V^*_{conc}. This may seem strange when we instantiate V^*_{base} with the random termination verifier V^{k*} verifier because, as we recall, the random termination verifier needs to terminate with fresh randomness, even when rewound. There is no contradiction here, however. Recall that V^{k*} simulates this fresh randomness also with the use of hash-functions; therefore, when V^*_{base} is instantiated with V^{k*}, the randomness of V^*_{base} actually includes a hash-function. This modular treatment — freshly generating hash-functions for each incarnation of V^{k*} — differs slightly from the more global definition of V^*_{conc} by [CKPR01] that shares one universal hash-function among all incarnations of V^{k*}.

[7] Equivalent, but more formally, a query τ is *inside* $V^*_{\text{conc}}(n, g, d)$ if τ consists of a partial and *incomplete* transcript of $V^*_{\text{base}}(n, d)$ invoked by $V^*_{\text{conc}}(n, g, d)$.

Induction Hypothesis. If all accepting incarnations V^*_{base} responds to at least q distinct initiating queries, then for every accepting incarnation $V^*_{\text{base}}(n, r)$ at depth d (invoked by some $V^*_{\text{conc}}(n, g, d)$), then V^*_{conc} must respond to at least

$$q^{d_{\max}-d+1}$$

distinct initiating queries that is inside $V^*_{\text{conc}}(n, g, d)$.

The base case is $d = d_{\max}$, which trivially follows by the premise that all accepting incarnations V^*_{base} responds to at least q distinct initiating queries. In general, suppose the induction holds for $d + 1$, we show that the induction holds for d.

Let $V^*_{\text{base}}(n, r)$ be an accepting incarnation at depth d invoked by some $V^*_{\text{conc}}(n, g, d)$. By the premise, $V^*_{\text{base}}(n, r)$ responds to at least q distinct initiating queries $\tau_1, \ldots, \tau_q$, where each τ_i is associated with a recursive call $V^*_{\text{conc}}(n, g, d+1)$ and a corresponding invoked incarnation $V^*_{\text{base}}(n, r_i)$. By the induction hypothesis, V^*_{conc} must respond to at least $q^{d_{\max}-d}$ distinct initiating queries inside $V^*_{\text{conc}}(n, g, d + 1)$, for each of q calls $V^*_{\text{conc}}(n, g, d + 1)$. Note that all these queries are inside $V^*_{\text{conc}}(n, g, d)$ by definition, and the queries inside different calls of $V^*_{\text{conc}}(n, g, d+1)$ must be distinct. Therefore, V^*_{conc} responds to at least $q^{d_{\max}-d+1}$ distinct initiating queries inside $V^*_{\text{conc}}(n, g, d)$. This completes the proof of the induction.

Finally, if $S^{V^*_{\text{conc}}}$ outputs an accepting view of V^*_{conc}, then $S^{V^*_{\text{conc}}}$ contains an accepting incarnation of $V^*_{\text{base}}(n, r)$ of $V^*_{\text{conc}}(n, g, 1)$. By the above induction, V^*_{conc} must respond to at least $q^{d_{\max}}$ distinct initiating queries during the interaction, which implies that S must make at least $q^{d_{\max}}$ queries to V^*_{conc}. □

In the next section, we will use Lemma 9 in a contrapositive form stated as follows.

Corollary 10. *Let $q(n) \geq m$ be a parameter. If $S^{V^*_{conc}}$ outputs an accepting view of V^*_{conc} and makes less than $q^{d_{\max}}$ queries to V^*_{conc}, then during the execution of $S^{V^*_{conc}}$, there must exist an accepting incarnations V^*_{base} of V^*_{conc} that responds to less than q distinct initiating queries.*

5.3 A Reduction Lemma and the Soundness of $\mathcal{D}$

In this section, we prove the following "soundness" reduction lemma, from which the soundness of the decision procedure $\mathcal{D}$ follows immediately by instantiating V^*_{base} with the random termination verifier V^{k*} constructed in Section 4.1, and applying Lemma 7.

Lemma 11. *Let $q(n) \geq m^{\omega(1)}$ be a parameter. Let V^*_{base} be a concurrent verifier for Π that schedules in m rounds. Let V^*_{conc} be the concurrent verifier constructed in Section 5.1 using V^*_{base} as a base verifier. Let $x \notin L$ be an input. If for every oracle* PPT *machine S_{base} that makes at most q queries to the oracle, $S^{V^*_{base}}_{base}$ can only output an accepting view of V^*_{base} with negligible probability, then for every oracle* PPT *machine S, $S^{V^*_{conc}}$ can only output an accepting view of V^*_{conc} with negligible probability.*

Proof. Suppose for the sake of contradiction that there exists an $x \notin L$ such that $S^{V^*_{\text{conc}}}(x)$ outputs an accepting view of V^*_{conc} with some noticeable probability ε. We shall construct, from S, an oracle PPT machine S_{base} such that $S^{V^{n*}}_{base}$ makes at most q queries to V^*_{base} and outputs an accepting view of V^*_{base} with noticeable probability, contradicting to the premise of the lemma.

At a high level, S_{base} simply embeds V^*_{base} in the interaction of S and V^*_{conc} as an incarnation of V^*_{base}. For clarity, we denote the *external* oracle verifier (given to S_{base}) by V^{*ext}_{base}. Let $T(n)$ be a time bound of S. We construct S_{base} as follows.

$S_{base}^{V^{n*}}$ chooses a random $i \leftarrow [T]$, and simulates the interaction of S and V^*_{conc} in his head, with V^{*ext}_{base} plays the role of the i-th invoked incarnation V^*_{base} (invoked by some $V^*_{\text{conc}}(n, g, d)$). More precisely,

- When V^*_{base} is invoked, V^*_{base} needs to send the first verifier message. S_{base} uses the first message of V^{*ext}_{base} to simulate V^*_{base}'s first message.
- When V^*_{base} needs to respond to a prover query τ,[8] S_{base} forwards the query to V^{*ext}_{base} and use it's answer to simulate V^*_{base}.

  Note that V^*_{base} may need to respond to the same prover query multiple times,[9] with *the same* response. Hence, S_{base} only forwards the query once, and simulate the extra ones with the same response *without* forwarding.
- The remaining case is when S makes a closing query to $V^*_{\text{conc}}(n, g, d)$. In this case, S_{base} does *not* forward the query out. Note that we assume (w.l.o.g.) that the decision of V^*_{base} can be computed from a complete transcription, so S_{base} can compute whether $V^*_{\text{conc}}(n, g, d)$ accepts or not and continue the simulation.
- Finally, if S_{base} needs to forward more than q queries, then S_{base} simply aborts. If S_{base} finished the simulation of $S^{V^*_{\text{conc}}}$, then S_{base} outputs an accepting view of V^{*ext}_{base} if such a view exists, i.e., if V^*_{base} is an accepting incarnation.

Note that by construction, S_{base} forward exactly one query for every initiating queries responded by $V^*_{\text{conc}}(n, g, d)$. Also note that since S runs in $T(n) \leq \text{poly}(n)$ time, S can makes at most

$$T < q^{d_{\max}} = q^{\log_m n} = n^{\omega(1)}$$

queries to V^*_{conc}. By Corollary 10, when S outputs an accepting view of V^*_{conc}, there must exist an accepting incarnation V^*_{base} of $S^{V^*_{\text{conc}}}$ that responds to at most q queries (referred to as a *good* incarnation). Finally, note that S_{base} perfectly simulates $S^{V^*_{\text{conc}}}$ whenever S_{base} chooses a good incarnation. Therefore,

$$\begin{aligned}
&\Pr[\ S_{base}^{V^*_{\text{base}}} \text{ outputs an accepting view }] \\
&\geq \Pr[(\ S^{V^*_{\text{conc}}} \text{ outputs an accepting view }) \wedge (\ S_{base} \text{ chooses a good incarnation in } S^{V^*_{\text{conc}}}\ )] \\
&\geq \varepsilon/T,
\end{aligned}$$

contradicting to the premise of the lemma. □

Finally, for the soundness of the decision procedure $\mathcal{D}$, consider V^*_{conc} with V^*_{base} instantiated by the random termination verifier V^{k*} constructed in Section 4.1, with k set to be n. Let $q = n^{1/m}/\log^2 n$, and note that when $m \in o(\log n/\log\log n)$,

$$q = n^{1/m}/\log^2 n \geq m^{\omega(1)}.$$

The soundness of $\mathcal{D}$ follows by composing Lemma 11 and (the generalized version of) Lemma 7 with the above parameters. This completes the proof of Theorem 8.

[8] Recall that τ is an initiating query associated with a recursive call $V^*_{\text{conc}}(n, g, d+1)$, and V^*_{base} needs to respond to τ when S makes a closing query τ' to $V^*_{\text{conc}}(n, g, d+1)$ and makes $V^*_{\text{conc}}(n, g, d+1)$ accept.

[9] This happens when S makes multiple closing queries τ''s to $V^*_{\text{conc}}(n, g, d+1)$.

5.4 Quasi-polynomial Time Simulation Lower Bound for $O(1)$-round cZK

We end the section with a concrete application of our modular framework: we present a new lower bound that essentially shows that constant-round, black-box cZK protocols must employ quasi-polynomial time simulators.

Theorem 12. *Let n be the security parameter and L be a language. Let $T \in n^{o(\log n)}$ (e.g., $T(n) = n^{\log^{0.99} n}$), and let $\mathrm{BPT}(\mathrm{poly}(T))$ be class of bounded probabilistic time machines that run in time $\mathrm{poly}(T)$. Suppose Π is a constant-round black-box concurrent zero-knowledge argument for L against $\mathrm{BPT}(\mathrm{poly}(T))$ malicious verifier. Further assume that Π has perfect completeness, and soundness error $\mathsf{ngl}(\mathsf{T}) = 1/T^{\omega(1)}$ against cheating provers in $\mathrm{BPT}(\mathrm{poly}(T))$. Suppose the black-box simulator S of Π runs in time $\mathrm{poly}(T)$. Then L is decidable by $\mathrm{BPT}(\mathrm{poly}(T))$.*

Proof Sketch. Let Π have m rounds (a constant). We start the proof in the same manner as Theorem 2 and 8, and omit many of the repeating details. Define V^*_{conc} as in Section 5.1, and again instantiate V^*_{base} with the random termination verifier V^{k*} with $k = n$. Our main goal is to show that $\mathcal{D}$ (similarly defined) is sound, i.e., on inputs $S^{V^*_{\mathrm{conc}}}(x)$, S cannot produce an accepting transcript of V^*_{conc}.

We start with Lemma 7, which states that any oracle machine given black-box access to V^*_{base} cannot output an accepting view of V^*_{base} unless it makes

$$> n^{1/m}/\log^2 n \quad \text{queries.}$$

By apply Lemma 9 to amplify this query lower bound, we have that any oracle machine given black-box access to V^*_{conc} cannot output an accepting view of V^*_{conc} unless it makes

$$> (n^{1/m}/\log^2 n)^{\log_m n} \in n^{\Omega(\log n)} \quad \text{queries}$$

(formally a reduction similar to Section 5.3 is required). Here we have used the fact that m is a constant. But the black-box simulator S runs in time $\mathrm{poly}(T) \in o(n^{O(\log n)})$, and thus can never afford to make so many queries! Therefore S cannot produce an accepting transcript of V^*_{conc}. □

We make a few remarks about Theorem 12 (first about the theorem itself, and then about some of the omitted proof details):

- As an upper bound, Pass and Venkitasubramaniam [PV08] construct constant-round black-box concurrent zero-knowledge protocols for all of NP in the model where both the simulator and the adversarial verifier runs in time $n^{\mathrm{poly}(\log n)}$. In comparison, Theorem 12, on the other hand, shows that such a result would be impossible for time complexity less than $n^{o(\log n)}$.

- A closely related lower bound by Rosen [Ros00] (a predecessor of [CKPR01]) showed that a (polynomial) black-box simulator of 7-round concurrent zero-knowledge protocol for non-trivial languages must make $n^{O(\log n/\log\log n)}$ queries (i.e., a contradiction). Our results are stronger in that they work for all constant round protocols and and yield an improver query-complexity lower bound. On the other hand, Rosen's lower bound is stronger in that it applies also to non-aborting verifiers.

- Note that because the simulator S now is super-polynomial time, we require that the original protocol's soundness to hold against super polynomial time machines as well (so that S is cannot break soundness), and our conclusion is weakened to $L \in \mathrm{BPT}(\mathrm{poly}(T))$ (since the decision procedure $\mathcal{D}$ incorporates S).

Additionally, note that we need the class of adversarial verifiers for which the zero-knowledge property holds to contain super-polynomial time algorithms. The reason for this is the following. Recall that the adversarial verifier must generate "fresh randomness" in the face of a rewinding simulator. Since the simulator may run in super-polynomial time, we need to employ hash-functions are t-wise independent for a super-polynomial t, and such hash function require super-polynomial description length.

Note, however. that if we assume the existence of PRFs secure against quasi-polynomial-time, then the lower bound applies also to polynomial-time verifier: Simply replace the t-wise independent hash-function with such a PRF.

Acknowledgments

We thank to Iftach Haitner and Johan Håstad for useful discussion in the early stage of this research.

References

[Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS '01*, pages 106–115, 2001.

[BG02] Boaz Barak and Oded Goldreich. Universal arguments and their applications. In *Computational Complexity*, pages 162–171, 2002.

[BL02] Boaz Barak and Yehuda Lindell. Strict polynomial-time in simulation and extraction. In *STOC '02*, pages 484–493, 2002.

[Blu86] M. Blum. How to prove a theorem so no one else can claim it. *Proc. of the International Congress of Mathematicians*, pages 1444–1451, 1986.

[CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO '94*, pages 174–187, 1994.

[CKPR01] Ran Canetti, Joe Kilian, Erez Petrank, and Alon Rosen. Black-box concurrent zero-knowledge requires $\tilde{\omega}(\log n)$ rounds. In *STOC '01*, pages 570–579, 2001.

[DNS04] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. *J. ACM*, 51(6):851–898, 2004.

[FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO*, pages 526–544, 1989.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *STOC*, pages 416–426, 1990.

[GK96a] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[GK96b] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, 1996.

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

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

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7:1–32, 1994.

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

[Gol02] Oded Goldreich. Concurrent zero-knowledge with timing, revisited. In *STOC '02*, pages 332–340, 2002.

[Hai09] Iftach Haitner. A parallel repetition theorem for any interactive argument. In *FOCS '09*, pages 241–250, 2009.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28:12–24, 1999.

[HRS09] Iftach Haitner, Alon Rosen, and Ronen Shaltiel. On the (im)possibility of Arthur-Merlin witness hiding protocols. In *TCC '09*, pages 220–237, 2009.

[Kat08] Jonathan Katz. Which languages have 4-round zero-knowledge proofs? In *Theory of Cryptography*, pages 73–88, 2008.

[KPR98] Joe Kilian, Erez Petrank, and Charles Rackoff. Lower bounds for zero knowledge on the internet. In *FOCS '98*, pages 484–492, 1998.

[MP06] Silvio Micali and Rafael Pass. Local zero knowledge. In *STOC '06*, pages 306–315, 2006.

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

[PPS+08] Omkant Pandey, Rafael Pass, Amit Sahai, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Precise concurrent zero knowledge. In *EUROCRYPT '08*, pages 397–414, 2008.

[PTV10] Rafael Pass, Wei-Lung Dustin Tseng, and Muthuramakrishnan Venkitasubramaniam. Eye for an eye: Efficient concurrent zero-knowledge in the timing model. In *TCC*, pages 518–534, 2010.

[PTW09] Rafael Pass, Wei-Lung Dustin Tseng, and Douglas Wikström. On the composition of public-coin zero-knowledge protocols. In *CRYPTO '09*, pages 160–176, 2009.

[PV08] Rafael Pass and Muthuramakrishnan Venkitasubramaniam. On constant-round concurrent zero-knowledge. In *TCC '08*, pages 553–570, 2008.

[Ros00] Alon Rosen. A note on the round-complexity of concurrent zero-knowledge. In *CRYPTO '00*, pages 451–468, 2000.