On the Composition of Public-Coin Zero-Knowledge Protocols

Rafael Pass[1], Wei-Lung Dustin Tseng[1], and Douglas Wikström[2]

[1] Cornell University, USA
[2] KTH Royal Institute of Technology, Sweden

Abstract. We show that only languages in BPP have public-coin, black-box zero-knowledge protocols that are secure under an unbounded (polynomial) number of parallel repetitions. This result holds both in the plain model (without any set-up) and in the Bare Public-Key Model (where the prover and the verifier have registered public keys). We complement this result by showing the existence of a public-coin black-box zero-knowledge proof that remains secure under any *a-priori* bounded number of concurrent executions.

1 Introduction

Zero-knowledge (ZK) interactive protocols [GMR89] are paradoxical constructs that allow one player P (called the prover) to convince another player V (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, ZK proofs have numerous cryptographic applications and are one of the most fundamental cryptographic building blocks. A fundamental question regarding zero-knowledge protocols is whether their composition remains zero-knowledge. The three most basic notions of compositions are sequential composition [GMR89, GO94], parallel composition [FS90, GK96b] and concurrent composition [FS90, DNS04]. In a sequential composition, the players sequentially run many instances of a zero-knowledge protocol, one after the other. In a parallel composition, the instances instead proceed in parallel, at the same pace. Finally, in a concurrent composition, messages from different instances of the protocol may be arbitrarily interleaved.

While the definition of ZK is closed under sequential composition [GO94], this no longer holds for parallel composition [GK96b] (and thus not for concurrent composition either). However, there are zero-knowledge protocols for all of NP that have been demonstrated to be secure under both parallel and concurrent composition. For the case of only parallel composition, constant-round protocols are known [Gol02, FS90, GK96a]. For the case of concurrent composition, a series of work [RK99, KP01, PRS02] show feasibility of $\tilde{O}(\log n)$-round protocols; furthermore, this round-complexity is essentially optimal with respect to black-box simulation [KPR98, Ros00, CKPR01].

Whereas the original ZK protocols of [GMR89, GMW91, Blu87] are *public-coin*—i.e., the verifier's messages are its random coin-tosses—all of the aforementioned parallel or concurrent ZK protocols use private coins. Indeed, in their

S. Halevi (Ed.): CRYPTO 2009, LNCS 5677, pp. 160–176, 2009.

seminal paper, Goldreich and Krawczyk [GK96b] (GK from now on) showed that only languages in BPP have *constant-round* public-coin (stand-alone) black-box ZK protocols with negligible soundness error, let alone the question of parallel composition. In particular, their results imply that (unless NP $\subseteq$ BPP) the constant-round ZK protocols of e.g., [GMW91, Blu87] with constant soundness error cannot be black-box ZK under parallel repetition (as this would yield a constant-round black-box ZK protocol with negligible soundness error).

A natural question is whether the constant-round restriction imposed by the GK result is necessary. Namely,

> *Is there a (possibly super-constant round) public-coin black-box ZK protocol that is secure under parallel (or even concurrent) composition?*

Our results. In this work, we provide a negative answer to the above question. Namely, we show that only languages in BPP have public-coin black-box ZK protocols that remain secure under parallel (and thus also concurrent) composition. Thus, whereas for private-coin protocols, a super constant number of rounds helps in establishing concurrent composition [RK99, KP01, PRS02], we conclude that it is not the case for public-coin protocols.

Theorem (Informal). *If L has a public-coin argument that is black-box parallel ZK, then $L \in$ BPP.*

In fact, our result establishes that any black-box ZK protocol that remains secure under m parallel executions must have $\tilde{\Omega}(m^{1/2})$ rounds.

On the positive side we show that every language in NP has a public-coin black-box ZK proof that remains secure under an *a-priori* bounded number of concurrent (and thus parallel) executions.

Theorem (Informal). *Assume the existence of one-way functions. Then for every polynomial m, there exists an $O(m^3)$-round public-coin black-box m-bounded concurrent ZK proof for NP.*

This complements a result of [Bar01], which constructs constant-round public-coin bounded-concurrent ZK *arguments* (rather than proofs) using *non-black-box* simulation.

We next turn to compositions in models with trusted set-up. Canetti, Goldreich, Goldwasser and Micali [CGGM00] show that in the Bare Public-Key (BPK) Model, where each player has a registered public-key, constant-round black-box concurrent ZK protocols exist for all of NP (whereas in the plain model without set-up, as mentioned above, $\tilde{\Omega}(\log n)$ rounds are necessary for non-trivial languages [CKPR01]). We show that for the case of public-coin protocols, the BPK set-up does not help with composition.

Theorem (Informal). *If L has a public-coin argument in the BPK model that is black-box parallel ZK, then $L \in$ BPP.*

Finally, as we will see, some of the intermediate ideas in our work are closely related to the notion of *resettable soundness* [BGGL01]. Very informally, we

establish that parallel repetition of public-coin protocols not only reduces the soundness error [PV07, HPPW08], but also *qualitatively* strengthen the soundness—the new protocols will be secure even under a "resetting" attack.

Techniques. To describe our technique, let us first briefly recall the GK lower bound showing that only languages in L have $O(1)$-round public-coin black-box ZK. Let (P, V) be a black-box ZK proof of a language L. Consider a malicious verifier V^* which, instead of picking its messages at random, computes them by applying a hash function to the current transcript. GK show that any black-box simulator S, together with V^* can decide L: on input x, simply run $S^{V^*(x)}(x)$ and accept if S outputs a view where V^* is accepting. It easily follows that if $x \in L$, then $S^{V^*}(x)$ will output a transcript where V^* is accepting (as the honest prover would convince V^*). The crux of their proof is to show that if $x \notin L$, then $S^{V^*}(x)$ will output an accepting view only with small probability. If S was not rewinding V^* this would directly follow from the soundness pf (P, V). However S might rewind V^*, and might only convince V^* in one of its rewinding "threads". Nonetheless, GK manages to show that if S—using rewinding, or "resetting"—manages to convince V^*, then we can construct a machine T that uses S to convince an external verifier V (without rewinding), contradicting the soundness of (P, V). In other words, they show that the protocol (P, V^*) is sound under a "resetting-attack" [CGGM00, BGGL01]. Analogously, to prove our results, we show that if we have take a public-coin interactive proof (P, V) and repeat it sufficiently many times in parallel (and again letting the verifier pick its messages by applying a hash function to the transcript), then the resulting protocol is sound under a resetting-attack.

More details on the reduction. GK, as well as all subsequent black-box lower bounds e.g., ([KPR98, Ros00, CKPR01, BL02, Kat08, HRS09]) rely on the following approach for constructing the stand-alone (non-resetting) prover T (given the "rewinding" simulator S). T incorporates S and internally emulates the execution of S with an internally emulated verifier (which of course can be rewound). While doing this emulation, T also appropriately picks some messages sent by S to the internal verifier, and forwards them externally (and also forwards back the reply received externally). The crux of the various lower bounds is how these messages (to be forwarded externally) are chosen. The difficulty of this task stems from the fact that, at the time of deciding whether to externally forward a message or not, T does not yet know if the simulator will eventually choose this message to "continue" its simulation, or treat this messages simply as a "rewinding" (used to collect information).

For the case of constant-round protocols, GK show that a *random* selection of messages to forward externally works. (If the protocol has d rounds, this random selection is "correct" with probability $1/m^d$, where m is the number of queries made by the simulator to its verifier.) To handle a super constant number of rounds, Canetti, Kilian, Petrank and Rosen [CKPR01] show that when dealing with an adversarial verifier that can schedule messages in an arbitrary way, there exists some particular scheduling which makes it easy to identify appropriate messages to forward externally (as long as the number of rounds is sub-logarthmic).

This general approach of simply running the simulator S "straight-line" seems hard to extend to protocols with a polynomial number of rounds; the number of possible choices for messages to forward to the external verifier becomes too large. To get around this problem, we use a different technique. Instead of simply running S "straight-line", we let T *rewind* S (while S itself believes it is rewinding the verifier). This makes it possible for us to "check" whether a message is "good" before forwarding it to the external verifier. Our strategy is twofold. First, we only externally forward queries that have a good chance of being included in the output view; since the protocol is public-coin, we can estimate this chance by doing "test-runs" of S. Once we have forwarded a query, we will force S to include it in the output view by repeatedly rewinding S. Intuitively, if the forwarded queries are indeed "good", then T should break the soundness of Π after polynomially many rewinds. But what if the external verifier acts differently from our test-runs and replies with a "bad" response to a forwarded query? Using a probabilistic lemma due to Ran Raz [Raz98] (used to prove that parallel repetition reduces the soundness error in two-prover games) we can show that if we have enough parallel sessions, and the external verifier only decides the verifier response in one session, the "goodness" of a forwarded query will not change much.

We remark that our approach shares similarities with previous works on the topic of soundness amplification under parallel repetitions, such as [BIN97] [PV07] [IJK07], and especially [HPPW08]; in particular, our use of Raz's lemma is similar to its use in [HPPW08]. However, whereas those works show how to transform a parallel prover with "small" success probability into a stand-alone prover with "high" success probability, we show how to transform a *rewinding* parallel prover into a (non-rewinding) stand-alone prover. This requires overcoming several novel obstacles: most notably, we are required to deal with the difficulty of forcing the simulator S to output a view which uses the queries externally forwarded by T.

To extend our lower bound to the BPK model, we run into the additional problem that the external verifier can decide whether to accept or reject based on its secret key (which T does not know). T can thus no longer determine whether an external verifier accepts or rejects when doing test-runs, which is crucial for deciding which messages to forward externally. By relying on the "trust-halving" technique from [IW97, BIN97], and its refinement in [HPPW08], we show how T can make an "educated" guess which is sufficiently good.

Parallel-repetition and Resettable-soundness. As an independent contribution, we believe that our techniques elucidates an intriguing (and useful) connection between *lower bounds* for black-box ZK, and feasibility results for soundness/hardness amplification. As mentioned, our core technical contribution shows that (appropriate) parallel-repetition of a public-coin protocol not only reduced the soundness error [PV07, HPPW08], but also yields a protocol that is sound even under a resetting attack [GK96b, CGGM00, BGGL01]. To establish our ZK lowerbound, we only consider a "weak" notion of "resettable-soundness" where the statement to be proved cannot be changed. In the full version of the paper, we shows that if the original protocol also is a *proof of*

knowledge [GMR89, FS90, BG02], then the parallelized version also satisfies the stronger notion of resettable-soundness from [BGGL01] (where the adversary can also change the statement during its rewindings). [BGGL01] showed a similar type of result for $O(1)$-round public-coin proofs of knowledge.

Outline. We introduce some preliminaries in Sect. 2, and jump into our impossibility results in Sect. 3. Next we present our public-coin bounded-concurrent zero-knowledge protocol in Sect. 4. Details of our extension to the Bare Public Key model and our application to resettable soundness can be found in the full version of this paper.

2 Preliminaries

We assume familiarity with indistinguishability, interactive proofs and commitments. $|x|$ denotes the length of a (bit) string x, and $[n]$ denotes the set $\{1, \ldots, n\}$.

Let $\Pi = \langle P, V \rangle$ be an interactive proof for a language L between prover P and verifier V. 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 alternatively. The notation $\langle v_1, p_1, \ldots \rangle$ specifies a full or partial **transcript** of Π where v denotes verifier messages and p denotes prover messages. Π is **public-coin** if the verifier messages are just independent segments of V's random tape.

We may repeat an interactive proof in parallel. Let $\Pi^m = \langle P^m, V^m \rangle$ be Π repeated in m **parallel sessions**; that is, each prover and verifier message in Π^m is just concatenation of m copies of the corresponding message in Π. V^m completes Π in all m sessions (or abort in all sessions), and accepts if and only if all m sessions are accepted by V.

In general, an adversarial verifier is not restricted to parallel schedules. 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**). 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^*). *Note that for public-coin protocols, the view of V^* is just the transcript of the interaction.*

A **black-box simulator** S is a probabilistic polynomial time machine that is given black-box access to V^* (written as $S = S^{V^*}$). Formally, the random tape of V^*, denoted by r, is uniformly chosen and fixed a priori, and S is allowed to specify a valid partial transcript $\tau = \langle v_1, p_1, \ldots, p_i \rangle$ of $\langle P, V^*_r \rangle$, 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^* would have responded given the previous prover messages $p_1, \ldots, p_{j-1}$ (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) adversarial verifier V^* can be generated by a simulator. The

protocol is concurrent ZK 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, random tape r, and every (stand-alone) adversary V^*, $S^{V_r^*(x,z)}(x)$ runs in time polynomial in $|x|$. Furthermore, the ensembles $\{\text{View}^P_{V_r^*}(x,z)\}_{x\in L, z, r\in\{0,1\}^*}$ and $\{S^{V_r^*(x,z)}(x)\}_{x\in L,z,r\in\{0,1\}^*}$ are computationally indistinguishable over $x \in L$.*

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 all polynomials m, there exists a black-box simulator S_m such that for every common input x, auxiliary input z, random tape r and every m-session concurrent adversary V^*, $S_m^{V_r^*(x,z)}(x)$ runs in time polynomial in $|x|$. Furthermore, the ensembles $\{\text{View}^P_{V_r^*}(x,z)\}_{x\in L,z,r\in\{0,1\}^*}$ and $\{S_m^{V_r^*(x,z)}(x)\}_{x\in L,z,r\in\{0,1\}^*}$ are computationally indistinguishable over $x \in L$.*

We also consider a bounded version of concurrent zero-knowledge where the order of quantifiers are reversed [Bar01].

Definition 3 (Black-Box Bounded Concurrent Zero-Knowledge). *Let $\Pi = \langle P, V \rangle$ be an interactive proof (or argument) for a language L and let m be a polynomial. Π is black-box m-bounded concurrent zero-knowledge if there exists a black-box simulator S such that for every common input x, auxiliary input z, random tape r, and every m-session concurrent adversary V^*, $S^{V_r^*(x,z)}(x)$ runs in time polynomial in $|x|$. Furthermore, the ensembles $\{\text{View}^P_{V_r^*}(x,z)\}_{x\in L,z,r\in\{0,1\}^*}$ and $\{S^{V_r^*(x,z)}(x)\}_{x\in L,z,r\in\{0,1\}^*}$ are computationally indistinguishable over $x \in L$.*

3 Impossibility

From now on zero-knowledge refers to *black-box* zero-knowledge. In this section we show that only languages in BPP have public-coin concurrent zero-knowledge protocols. We actually show a stronger result: Except for languages in BPP, no public-coin protocols remains black-box zero-knowledge when repeated in parallel. The formal theorems are stated below, where n denote the security parameter or the input size.

Theorem 1. *Suppose language L has a $k = \text{poly}(n)$-round public coin black-box zero-knowledge proof Π with soundness error $1/2$. If $m \geq k \log^2 n$ and Π^m is zero-knowledge, then $L \in$ BPP.*

Theorem 2. *Suppose language L has a $k = \text{poly}(n)$-round public-coin black-box zero-knowledge argument Π with soundness error $1/2$. If $m \geq (k^2 \log k) \log^2 n$ and Π^m is zero-knowledge, then $L \in$ BPP.*

We remark here that our theorems also hold with respect to so-called non-aborting verifiers that never send an invalid message.

3.1 Common Proof Components

The proofs of Theorem 1 and 2 begin in the same high-level framework as that of [GK96b]. Suppose a language L has a public-coin ZK protocol $\Pi = \langle P, V \rangle$, and Π^m is zero-knowledge with a black-box simulator S that runs in time n^d. To show that $L \in \mathsf{BPP}$, we construct a "random-looking" adversarial verifier, V^*, and consider the following decision algorithm D: $D(x)$ runs S^{V^*} to generate a view of V^*, and accepts x if and only if V^* accepts given the generated view (which in turn occurs if and only if the honest verifier V accepts in all m sessions of the view).

V^* is actually a family of adversarial verifiers constructed as follows. Let H be a family of hash functions that is random enough compared to the running time of S; formally, H should be n^d-wise independent (see [GK96b, CG89]). Let V_h^* be the verifier that when queried with transcript τ, responds (deterministically) with the message $h(\tau)$. We write V^* to mean V_h^* for a randomly chosen h, i.e., when D runs S^{V^*}, D first chooses h randomly from H and then run $S^{V_h^*}$.

We make two easy observations about S^{V^*}. First, we may assume that whenever S queries V^* with a transcript or outputs a transcript τ, it first queries V^* with all the prefixes of τ; this only increases the running time of S polynomially. Second, we may assume that S never queries V^* with the same transcript twice (instead S may keep a table of answers). Then the set of all responses generated by V^* is identical to the uniform distribution since H is n^d-independent and S makes at most n^d queries to V^*.

We need to show that decision procedure D is both complete and sound. Completeness states that if $x \in L$, then D should accept x with probability at least $2/3$. This easily follows: The output of $S^{V^*}(x)$ is indistinguishable from the interaction of $\langle P^m, V^* \rangle$ since S is a zero-knowledge simulator. Furthermore, $\langle P^m, V^* \rangle$ is identical to m-copies of $\langle P, V \rangle$ since V^* produces independent, truly random verifier messages. Finally, by the completeness property of Π, V will accept x with probability 1 in all the copies of $\langle P, V \rangle$.

Soundness states that if $x \notin L$, then D should accept with probability at most $1/3$. That is, $S^{V^*}(x)$ can produce an accepting view of V^* with probability at most $1/3$. In a sense, this is the soundness of protocol Π^m against a rewinding prover such as S. This property is shown separately for proofs and arguments in the next two sections.

3.2 Proof of Theorem 1: Zero-Knowledge Proofs

We show that D is sound when Π is a proof. Our approach follows that of [GK96b] while relying on the soundness amplification theorem of [BM88].

Suppose for the sake of contradiction that for some $x \notin L$, $S^{V^*}(x)$ produces an accepting view with probability more than $1/3$; we will use S to lower bound the soundness error of Π^m (as an interactive proof of L). Whenever S^{V^*} outputs a valid accepting view of V^* and a corresponding transcript τ of Π^m, S would have queried V^* for each of the k verifier messages in τ (recall that we assumed this without loss of generality). A cheating prover of Π^m can therefore run S^{V^*} internally, guess which queries of S are used to form the accepting output

transcript, and forward them to an outside honest verifier of Π^m. Since S has maximum running time n^d, it can only query V^* for at most n^d messages. The probability of guessing all the right queries is then at least n^{-dk} (one guess for each round of Π). Note that forwarding queries to an outside honest verifier does not lower the acceptance probability of S^{V^*} since V^* is identical to a honest verifier (they both respond with random messages). Thus this cheating prover, using S, can break the soundness of Π^m with probability at least $(1/3)n^{-dk} = \Omega(2^{-dk \log n})$.

On the other hand, recall that Π has soundness error less than $1/2$. By the soundness amplification theorem of [BM88], Π^m should have soundness error at most $O(2^{-m})$. Since $m \geq k \log^2 n$, we have $O(2^{-m}) < \Omega(2^{-dk \log n})$ and reach a contradiction. □

3.3 Proof of Theorem 2: Zero-Knowledge Arguments

We now show that D is sound even when Π is an argument. Again we argue by contradiction, and suppose $S^{V^*}(x)$ outputs an accepting view for some $x \notin L$ with probably more than $1/3$. We cannot repeat the proof of Theorem 1 because parallel repetitions cannot reduce the soundness of arguments beyond being negligibly small. Therefore we cannot use S to break the soundness of Π^m. Instead, we directly show a parallel repetition theorem for "resettable-soundness"; that is, we relate the "resettable" soundness of $\langle P^m, V^* \rangle$ to the soundness of Π.

Proof Outline. The rest of this section describes how to construct a cheating prover T for Π. T runs S internally and plays the role of V^* when S makes a query. To break the soundness of Π, T needs to choose one of the m sessions and forward a complete set of S queries in that session (one for each round of Π) to an honest outside verifier V of Π. Moreover, S must eventually use these forwarded queries to output an accepting view. This is challenging since S may query V^* multiple times for each round of Π^m. While T must decide to forward a query or not at the time of the query, S can wait until all queries are completed before choosing which queries to form the output view. To overcome this obstacle, a key part of our analysis relies on *rewinding* S (note that at the same time, S believes that it is rewinding V^*). Our strategy is twofold. First we only forward queries that has some chance (preferably a good chance) of being included in the output view; this is done by doing test-runs of S. Once we have forwarded a query, we will force S to include it in the output view by repeatedly rewinding S.

We can describe a **transcript** of S as an alternating sequence of queries from S and responses from T, $[s_1, t_1, s_2, t_2, \ldots]$, where each S-query is in fact a partial transcript of Π^m that ends with a prover message (awaiting a verifier response). To avoid confusion, in our analysis τ and $\langle \cdot \rangle$ denote views of V^* (which are just transcripts of Π^m), while h and $[\cdot]$ denote transcripts of S. Recall that S may rewind V^*, and thus a transcript of S may be much longer than a view of V^*. Since the randomness of S is fixed, the behaviour of S is entirely determined by the T-responses in a transcript. The goal of T is then to generate a full transcript of S so that S produces an accepting view of V^* and a corresponding transcript

τ of Π^m, while simultaneously having the foresight to forward (a session of) all the S-queries pertaining to τ to the external verifier V (i.e. all S-queries that are a prefix of τ). If so, T has broken the soundness of Π, and we call this a **successful** simulation of S.

On a high level, T first fixes a random session $j_0 \in 1, \ldots, m$ as the forwarding session. Then, T will incrementally fix the transcript of S while forwarding S-queries to V in k iterations (one for each round of Π). During each iteration, T first forwards a query to V. Then, T continues the simulation of S using V's response until it finds a suitable query to forward in the next iteration. In more details:

Step 1. In iteration i, T starts with a partial transcript h_i of S that ends with a query for the i^{th} message of Π. T will forward session j_0 of this query to V and receive a reply $v_i^{j_0}$.

Step 2. Fixing the reply $v_i^{j_0}$, T randomly completes the partial transcript h_i up to $300k^2n^d$ times until it finds a successful completion h. If no successful completion is found, T aborts. Otherwise, let τ be the accepting view of V^* produced by S under transcript h (in particular, τ must include all the queries that have been forwarded by T). By assumption, S must query T for the $i+1^{\text{st}}$ verifier message in τ. Let h_{i+1} be the prefix of h up to this query (note that h_{i+1} is an extension of h_i), and this query (considered a "good" query for the $i+1^{\text{st}}$ verifier message) will be forwarded in the next iteration.

During the analysis, we first use Raz's lemma to show that because the number of sessions is large and j_0 was chosen randomly, we may pretend $v_i^{j_0}$ is nicely chosen conditioned on success, just like the other sessions (chosen by T in step 2). We also show that T rarely aborts.

Proof Details. We now introduce a series of hybrid simulators that formally defines and analyses T; all our hybrids will always generate truly random responses to S-queries so that S cannot distinguish the hybrids from V^*. We will start with a hypothetical hybrid, and gradually move towards T.

Hybrid 1. Our first hybrid $T^{(1)}$ serves to introduce the general idea of how T queries S internally; $T^{(1)}$ does not yet forward messages to the external verifier V.

$T^{(1)}$ builds a full transcript of S in $k+1$ iterations. In iteration i, $T^{(1)}$ fixes an S-query τ_i for the i^{th} message of Π^m. This query should have a good chance of being included by S in an accepting transcript of Π^m, and therefore is a good candidate to forward externally. Note that fixing an S-query amounts to fixing the transcript of S up until the desired S-query is made.

We now describe $T^{(1)}$ in detail. In the very beginning, $T^{(1)}$ fixes a random session $j_0 \in \{1, \ldots, m\}$; eventually the j_0^{th} session will be forwarded externally. After that, $T^{(1)}$ incrementally grows a transcript of S in k iterations. During the i^{th} iteration, $T^{(1)}$ receives a partial transcript of S from the previous iteration, $h_i = [t_1, s_1, \ldots, s_\ell = \tau_i]$, where τ_i is a S-query for the i^{th} verifier message of Π^m ($h_1 = []$, the empty transcript). Looking ahead, session j_0 of τ_i will be forwarded to the external V in later hybrids of T. As an invariant maintained by $T^{(1)}$, it should be possible to extend h_i into a full transcript of S where S outputs

an accepting view of V^* containing the query τ_i. We call such a full transcript a **successful** completion of h_i. Each iteration can be further divided into two steps:

Step 1. $T^{(1)}$ does not forward τ_i to the external V; instead it simulates a response to its liking. $T^{(1)}$ randomly samples a completion of h_i into h conditioned on success (always possible due to the invariant). Let $v_i^{(j_0)}$ be the response to τ_i in the j_0^{th} session in the successful completion h; it is used in place of a response from V. Let $\tilde{h}_i$ be a slight extension of the partial transcript h_i where the session j_0 response to τ_i is fixed to $v_i^{(j_0)}$.

Step 2. $T^{(1)}$ now samples a completion of $\tilde{h}_i$ into $\tilde{h}$ conditioned on success (note that h from the previous step is one such completion). Under $\tilde{h}$, S would output an accepting view τ of V^* (note that τ must extend τ_i). Let τ_{i+1} be the S query for the $i+1^{\text{st}}$ verifier message in τ (note that τ_{i+1} extends τ_i). $T^{(1)}$ then sets h_{i+1} to be the prefix of $\tilde{h}$ up to when S makes the query τ_i. Note that the invariant holds since by definition $\tilde{h}$ is a successful completion of h_{i+1}.

Note that in Step 2 of the final (k^{th}) iteration, $T^{(1)}$ simply outputs $\tilde{h}$ as a full transcript of S (there is no τ_{k+1} to fix). Due to the invariant, $T^{(1)}$ always produce a transcript of S where S outputs an accepting transcript τ, whose incremental prefixes $\tau_1, \ldots, \tau_k$ were forwarded by T to the external verifier.

Hybrid 2. Our second hybrid, $T^{(2)}$, describes a way to sample successful completions in Step 2 of each iteration (Step 1 will be replaced with the external verifier and is left alone for now). In Step 2, $T^{(2)}$ randomly completes the given partial execution ($\tilde{h}_i$) up to $300k^2n^d$ times, until a successful completion is found. If none of the completions are successful, $T^{(2)}$ aborts. Note that conditioned on $T^{(2)}$ not aborting, the output distribution of $T^{(2)}$ is *identical* to $T^{(1)}$.

To show that $T^{(2)}$ aborts with small probability, suppose for now that $T^{(2)}$ is allowed to sample an unbounded number of completions. Let us bound the expected number of completions that are needed to have a successful one. In the following analysis we distinguish between two probability spaces: $\Pr_S[\cdot]$ is used to measure probabilities over a single execution of S. On the other hand, $\Pr_T[\cdot]$ is used to measure probabilities over an execution of $T^{(2)}$ (with unbounded number of completions) which includes rewinding and executing S multiple times.

Let H_i and $\tilde{H}_i$ be the set of possible partial transcripts of S that is given to $T^{(2)}$ in Step 1 and Step 2 of the i^{th} iteration, respectively. Given $h \in H_i$ (or $\tilde{H}_i$), let $\Pr_S[h]$ denote the probability that a transcript of S has prefix h, and let $\Pr_T[h]$ denote the probability that $T^{(2)}$ is given h in the i^{th} iteration; similarly, $\Pr_S[\cdot \mid h]$ and $\Pr_T[\cdot \mid h]$ are probabilities conditioned on these events occurring. Let A^h be the event (over the S probability space) that a transcript of S has prefix h and is a successful completion of h; as a special case, $A = A^{\emptyset}$ is just the event that S outputs an accepting transcript. Also let R_i be the random variable (over the $T^{(2)}$ probability space) that denotes the number of completions performed by $T^{(2)}$ in step 2 of iteration i.

Lemma 3. $\mathbb{E}_T[R_i] \leq 3n^d$.

Proof. First expand $\mathbb{E}_T[R_i]$ by conditioning on the transcript h fixed in Step 1:

$$\mathbb{E}_T[R_i] = \sum_{h \in \tilde{H}_i} \Pr_T[h] \mathbb{E}_T[R_i \mid h] \tag{1}$$

Recall that in Step 2, $T^{(2)}$ samples random completions of h until a successful completion is found. Therefore

$$\mathbb{E}_T[R_i \mid h] = \frac{1}{\Pr_S[A^h \mid h]} \;\Rightarrow\; \mathbb{E}_T[R_i] = \sum_{h \in \tilde{H}_i} \Pr_T[h] \frac{1}{\Pr_S[A^h \mid h]} \tag{2}$$

We now state a claim and give its proof in the full version of this paper.

Claim 4. *Let* $h \in \tilde{H}_i$. $\Pr_T[h] \Pr_S[A] = \Pr_S[A^h]$.

Intuitively, the claim says that the probability of $T^{(2)}$ fixing h is proportional to the probability of successfully completing h ($\Pr_S[A]$, the probability that S produces an accepting transcript, is the normalizing factor). This can be shown by a counting argument. By expanding the RHS of Claim 4 and rearranging terms, we have

$$\begin{aligned} &\Pr_T[h] \Pr_S[A] = \Pr_S[A^h] = \Pr_S[h] \Pr_S[A^h \mid h] \\ \Rightarrow\; &\Pr_T[h] \frac{1}{\Pr_S[A^h \mid h]} = \Pr_S[h] \frac{1}{\Pr_S[A]} \leq 3 \Pr_S[h] \end{aligned}$$

since we assumed $\Pr_S[A] \geq 1/3$. Substituting this back into (2) gives

$$\mathbb{E}_T[R_i] \leq 3 \sum_{h \in \tilde{H}_i} \Pr_S[h] \tag{3}$$

finally, we may breakup the set $\tilde{H}_i$ based on the length of h which ranges from 1 to n^d (where length is the number of S-queries). Since each transcript of S has exactly one length ℓ prefix:

$$\mathbb{E}_T[R_i] \;\leq\; 3 \sum_{\ell=1}^{n^d} \sum_{h \in \tilde{H}_i, |h|=\ell} \Pr_S[h] \;\leq\; 3 \sum_{\ell=1}^{n^d} 1 = 3n^d$$

□

Now we can show that $300k^2n^d$ random completions are enough for $T^{(2)}$.

Lemma 5. $T^{(2)}$ *aborts with probability at most* $1/5$.

Proof. Since $\mathbb{E}_T[R_i] = \sum_{\tilde{h}_i} \Pr_T[\tilde{h}_i] \mathbb{E}_T[R_i \mid \tilde{h}_i] = \mathbb{E}_T[\mathbb{E}_T[R_i \mid \tilde{h}_i]] \leq 3n^d$, the Markov inequality states that the probability of $T^{(2)}$ fixing an $\tilde{h}_i$ such that $\mathbb{E}_T[R_i \mid \tilde{h}_i] \geq 30kn^d$ is at most $1/(10k)$. For each "good" $\tilde{h}_i$ where $\mathbb{E}_T[R_i \mid \tilde{h}_i] < 30kn^d$, we apply the Markov inequality again to obtain $\Pr_T[R_i \geq 300k^2n^d \mid h_i] \leq 1/(10k)$. Using the union bound we see that in any iteration, $T^{(2)}$ aborts in Step 1 with probability at most $1/(5k)$. A final union bound over k iterations of Step 2 shows that $T^{(2)}$ aborts overall with probability at most $1/5$. □

Hybrid 3. Our third and final hybrid $T^{(3)} = T$ differs from $T^{(2)}$ in Step 1 of each iteration. Recall that some session j_0 is chosen randomly as the forwarding session. Instead of generating $v_i^{(j_0)}$ in Step 1, $T^{(3)}$ asks the external honest verifier V for a verifier message. Because Π is public-coin, $T^{(3)}$ can continue to complete partial transcripts of S even if session j_0 is forwarded to V externally.

Given transcript $h_i = [t_1, s_1, \ldots, s_\ell = \tau_i]$ in iteration i, $T^{(3)}$ forwards session j_0 of τ_i to V, and uses the response from V as $v_i^{(j_0)}$ in Step 2. Suppose for now that $T^{(3)}$ does not abort and terminates successfully. Then S would have generated an accepting transcript τ of Π^m. Since $\tau_1, \ldots, \tau_k$ are prefixes of τ, session j_0 of τ would be an accepting transcript of Π consisting of forwarded prover messages and responses from V. This breaks the soundness of Π.

Therefore, it remains to show that $T^{(3)}$ is successful with probability more than 1/2. We will use Raz's lemma [Raz98] in analogy with [IJK07, HPPW08] to show that $v_i^{(j_0)}$ as generated by $T^{(1)}$ and $T^{(2)}$ is actually very close to the uniformly random messages generated by the honest verifier V. First we cite Raz's lemma as it appears in [Hol07, Lemma 5]:

Lemma 6. *Let $\{U_j\}_{j\in[m]}$ be independent random variables on $\mathcal{U}$ with probability distribution P_{U_j}. Let W be an event in $\mathcal{U}^m$ and $\Pr[W]$ be measured according to the joint probability distribution $\Pi_j P_{U_j}$. Then*

$$\sum_{j=1}^{m} \Delta(U_j|W, U_j) \leq \sqrt{m \log\left(\frac{1}{\Pr[W]}\right)}$$

where Δ is the statistical distance between distributions, and $U_j|W$ is the j^{th} component of an element in $\mathcal{U}^m$ chosen based on the joint probability distribution $\Pi_j P_{U_j}$, conditioned on W.

In other words, let $\{U_j\}_j$ be independent random variables, and let W be an event over $\Pi_j U_j$. If W occurs with high probability and there are many U_j, then on average over j, sampling U_j conditioned on W does not differ much from simply sampling U_j. Lemma 6 allows us the bound the change in success probability when $T^{(3)}$ forwards messages from a random session to V.

Lemma 7. *$T^{(3)}$ fails with probability at most $3/10 + O(1/\log n)$.*

Proof. We first construct a series of finer hybrids, $T_1, \ldots, T_{k+1}$, where T_i proceeds as $T^{(2)}$ until the start of iteration i (no forwarding), and continues as $T^{(3)}$ afterwards (with forwarding)[1]. Observe that $T_1 = T^{(3)}$ and $T_{k+1} = T^{(2)}$.

Consider two neighboring hybrids, T_i and T_{i+1}, which differ only in iteration i. Let h be the partial execution given in iteration i. For $j \in [m]$, let U_j be the random variable that denotes all the additional session j messages sent by T to randomly complete h, i.e., $\{U_j\}_j$ are independent and uniformly random. Let W^h be the event that the random messages $U_1, \ldots, U_m$ together produced

[1] This still makes sense since Π is a public-coin protocol; the outside verifier can directly generate a verifier response for any round of the protocol.

a successful completion of h. By definition, the distribution of $v_i^{(j_0)}$ produced by T_{i+1} (i.e., $T^{(2)}$) is just the first message of $U_{j_0}|W^h$. On the other hand, the distribution of $v_i^{(j_0)}$ produced by T_i (i.e., $T^{(3)}$) is just the uniform distribution, just like the first message of U_j.

Since T_{i-1} and T_i only differ in how $v_i^{(j)}$ is produced, their difference in success probability can be bounded by the statistical difference in the distributions of $v_i^{(j)}$. This is in turn bounded by:

$$\sum_{h\in H_i}\sum_{j=1}^{m}\Pr_T[h]\Pr[j_0=j]\Delta(U_j|W^h,U_j)=\sum_{h\in H_i}\Pr_T[h]\left(\frac{1}{m}\sum_{j=1}^{m}\Delta(U_j|W^h,U_j)\right)(*)$$

Lemma 6 states that for any event W,

$$\frac{1}{m}\sum_{j=1}^{m}\Delta(U_j|W,U_j)\le\sqrt{\frac{1}{m}\log\left(\frac{1}{\Pr[W]}\right)}$$

Observe that before iteration i, T_i and T_{i+1} are identical to $T^{(2)}$. When $T^{(2)}$ does not abort, $T^{(2)}$ is identical to $T^{(1)}$. In that case, Lemma 3 along with the Markov inequality implies that except with probability $1/(10k)$, $T^{(2)}$ fixes a "good" h with $\mathbb{E}_T[R_i \mid h]\le 30kn^d$, so that

$$\Pr[W^h]=\Pr_S[A^h \mid h]=\frac{1}{\mathbb{E}_T[R_i \mid h]}\ge\frac{1}{30kn^d}$$

We can now break the sum in (*) into two parts. Observe that

$$\sum_{\text{bad } h\in H_i}\Pr_T[h]\left(\frac{1}{m}\sum_{j=1}^{m}\Delta(U_j|W^h,U_j)\right)\le\sum_{\text{bad } h\in H_i}\Pr_T[h]\ \le\ \frac{1}{10k}$$

since statistical distances are bounded by 1, and

$$\begin{aligned}&\sum_{\text{good } h\in H_i}\Pr_T[h]\left(\frac{1}{m}\sum_{j=1}^{m}\Delta(U_j|W^h,U_j)\right)\\ &\le\sum_{\text{good } h\in H_i}\Pr_T[h]\sqrt{\frac{1}{m}\log(30kn^d)}\ \le\ \sqrt{\frac{1}{m}\log(30kn^d)}\end{aligned}$$

since $\sum_{h\in H_i}\Pr_T[h]=1$. Together, they show that (*) is at most

$$\frac{1}{10k}+\sqrt{\frac{1}{m}\log\left(30kn^d\right)}=\frac{1}{10k}+O\left(\frac{1}{k\log n}\right)$$

since $m\ge(k^2\log k)\log^2 n$. Summing up over the hybrids, and recalling that $T^{(2)}$ fails with probability at most $1/5$ (Lemma 5), $T^{(3)}$ fails with probability at most

$$\frac{1}{5}+k\left(\frac{1}{10k}+O\left(\frac{1}{k\log n}\right)\right)\le\frac{3}{10}+O\left(\frac{1}{\log n}\right)$$

□

Lemma 7 shows that T is successful with probability $> 1/2$, and completes the proof of Theorem 2. □

4 Bounded Concurrent Zero-Knowledge

In this section we give a family of public-coin proofs for NP, BoundedConcZK, parametrized by k, assuming the existence of one-way functions. The proof with parameter k has $2k^3 + 4$ rounds, and is k-bounded concurrent zero-knowledge whenever $k = \omega(\log n)$ where n is the input size.

4.1 A Bounded Concurrent Public-Coin ZK Protocol

Our construction of BoundedConcZK is similar in spirit to the concurrent zero-knowledge protocol of [RK99]. Given a language $L \in \mathsf{NP}$ and a parameter k, we construct a two stage public-coin proof $\langle P, V \rangle$ as follows. In stage one, $2k^3$ rounds of messages are exchanged where in each round, the prover gives a statistically binding commitment ([Nao91, HILL99]) of a random bit p_i, and the verifier responds with a random bit v_i; we call $p_i = v_i$ a **correct guess**. In stage two, $\langle P, V \rangle$ runs a 4-round public-coin witness indistinguishable proof of the modified NP statement "either $x \in L$ or that $p_i = v_i$ for $k^3 + k^2/2$ values of i", where x is the problem instance. This can be done, for example, by k parallel repetitions of the GMW 3-coloring protocol [GMW91]. The verifier accepts if the prover is successful with the stage two proof.

Protocol BoundedConcZK

Common Input: An instance x of a language $L \in \mathsf{NP}$ and a parameter k.
Stage One: For i from 1 to $2k^3$:
$P \to V$: Commit to a random bit p_i.
$V \to P$: Reply with a random bit v_i.
Stage Two: Run k parallel repetitions of the GMW 3-coloring protocol for the NP statement:

$$\Big(\text{there exists distinct } i_1, \ldots, i_{k^3 + \frac{1}{2}k^2} \text{ s.t. } p_{i_j} = v_{i_j} \text{ for all } j\Big) \vee (x \in L)$$

Fig. 1. Our public-coin black-box bounded concurrent zero-knowledge protocol

We set the round complexity of BoundedConcZK to $O(k^3)$ for the following two reasons. First, by the Chernoff bound, we expect that no adversarial prover can have more than $k^3 + O(\sqrt{k^3})$ correct guesses. Hence BoundedConcZK is sound. On the other hand, a zero-knowledge simulator can repeatedly rewind the verifier until it gets a correct guess. Intuitively (and shown formally later), in each round of stage one, the simulator can set one extra $p_i = v_i$ for *some* session, in addition to "natural luck" (that gives correct guesses for half of the sessions). Since the number of sessions is bounded by k, the simulator is able

to have $k^3 + O(k^3/k) = k^3 + O(k^2)$ correct guesses per session. This provides the simulator with a trapdoor to simulate stage two of the protocol, and hence BoundedConcZK is bounded concurrent zero-knowledge. We remark that k^3 was chosen for the sake of simplicity and is not optimized. The formal proof of completeness and soundness can be found in the full version of this paper.

4.2 Black-Box Bounded Concurrent Zero-Knowledge

We construct a black box simulator S such that given a malicious verifier, V^*, S^{V^*} generates the view of V^* in BoundedConcZK, provided that the number of concurrent sessions m satisfies $m \leq k$. The goal of S is to obtain as many correct guesses as possible by rewinding V^*. Towards that goal, S employs a simple greedy strategy to incrementally generate a partial view of V^*. Whenever V^* sends S a first stage message v_i, S checks if it had guessed correctly when committing to p_i. If so, S lengthens the partial view of V^* to include this correct guess. Otherwise, S rewinds V^* back to the previously generated partial view. This "incremental strategy" is somewhat reminiscent of [Lin03], but since our protocol is public-coin, the actual analysis is quite different.

We use superscripts to distinguish messages from different sessions. To prevent S from focusing too much on one particular session, we keep m counters, $c^1, \ldots, c^m$, to record how much "work" has been done in each session. In general, S proceeds as follows to incrementally fix the output (originally the empty view is fixed):

1. S commits to a fresh random bit for each stage one prover message.
2. For each stage two proof, S aborts if in this session, $p_i = v_i$ for less than $k^3 + k^2/2$ values of i. Otherwise, S uses this as a witness to complete the stage two proof.
3. If S receives a message v_i^j (from session j) and $c^j < 2k^2$, it checks if the commitment to p_i^j is part of the fixed output. If yes, then nothing can be done, so S simply continues. Otherwise, S checks if $p_i^j = v_i^j$. If yes, S takes the opportunity to fix the execution up to message v_i^j as part of the output and increments c^j; in this case we say v_i^j is *rigged*. If $p_i^j \neq v_i^j$, then S rewinds V^* to start a fresh continuation from the currently fixed output.
4. If S has performed $k-1$ rewinds without rigging a message, and on the k^{th} try again receives $v_i^j \neq p_i^j$ where p_i^j is not fixed and $c^j < 2k^2$, S simply gives up and pretend to *rig* v_i^j anyway (albeit incorrectly). That is, S fixes the output up to message v_i^j and increments c^j.

Claim 8. *S is a k-bounded black-box zero-knowledge simulator when $k \in \omega(\log n)$.*

Proof (sketch). We give a proof sketch here, and defer the full proof to the full version of this paper. Suppose for now that all p_i and v_i are independent and uniformly random (intuitively because the prover commitments are computationally hiding). We claim that except with negligible probability, S will have $k^3 + k^2/2$ correct guesses per session. If so, S can complete the stage two proofs (using the witness indistinguishable property) and generate the view of V^*.

To show our claim, observe that whenever a message is rigged, at most one commitment from each session is fixed to be part of the output, because before a second commitment appears in the same session, S would have tried to rig the first commitment first (unless this session already has $2k^2$ messages rigged). Since S rigs at most $2k^2$ messages from each session, and there are $2k^3$ messages and at most k sessions, every session will actually have $2k^2$ messages rigged.

Since all p_i and v_i are independent, a rigged message is always a correct guess except with probability 2^{-k}. Since there are $m(k^3+2k^2) \leq 2k^4$ messages in total, the union bound says that except with $2k^4 2^{-k}$ probability, all rigged messages are correct guesses. Next, for the $2k^3-2k^2$ messages that are not rigged, we apply the Chernoff bound to see that except with probability $e^{-k/4}$, we should have at least $(k^3 - k^2) - k^2/2$ correct guesses. Thus except with negligible probability, we have a total of $k^3 + k^2/2$ correct guesses as desired. □

Acknowledgements. We would like to thank Johan Håstad and the reviewers for invaluable comments, and for pointing out our work's connection with resettable soundness.

References

[Bar01] Barak, B.: How to go beyond the black-box simulation barrier. In: FOCS 2001, pp. 106–115 (2001)

[BG02] Barak, B., Goldreich, O.: Universal arguments and their applications. In: Computational Complexity, pp. 162–171 (2002)

[BGGL01] Barak, B., Goldreich, O., Goldwasser, S., Lindell, Y.: Resettably-sound zero-knowledge and its applications. In: FOCS 2002, pp. 116–125 (2001)

[BIN97] Bellare, M., Impagliazzo, R., Naor, M.: Does parallel repetition lower the error in computationally sound protocols? In: FOCS 1997, pp. 374–383 (1997)

[BL02] Barak, B., Lindell, Y.: Strict polynomial-time in simulation and extraction. In: STOC 2002, pp. 484–493 (2002)

[Blu87] Blum, M.: How to prove a theorem so no one else can claim it. In: Proceedings of the International Congress of Mathematicians, pp. 1444–1451 (1987)

[BM88] Babai, L., Moran, S.: Arthur-merlin games: a randomized proof system, and a hierarchy of complexity class. J. Comput. Syst. Sci. 36(2), 254–276 (1988)

[CG89] Chor, B., Goldreich, O.: On the power of two-point based sampling. J. Complex. 5(1), 96–106 (1989)

[CGGM00] Canetti, R., Goldreich, O., Goldwasser, S., Micali, S.: Resettable zero-knowledge (extended abstract). In: STOC 2000, pp. 235–244 (2000)

[CKPR01] Canetti, R., Kilian, J., Petrank, E., Rosen, A.: Black-box concurrent zero-knowledge requires $\tilde{\omega}(\log n)$ rounds. In: STOC 2001, pp. 570–579 (2001)

[DNS04] Dwork, C., Naor, M., Sahai, A.: Concurrent zero-knowledge. J. ACM 51(6), 851–898 (2004)

[FS90] Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: STOC 1990, pp. 416–426 (1990)

[GK96a] Goldreich, O., Kahan, A.: How to construct constant-round zero-knowledge proof systems for NP. Journal of Cryptology 9(3), 167–189 (1996)

[GK96b] Goldreich, O., Krawczyk, H.: On the composition of zero-knowledge proof systems. SICOMP 25(1), 169–192 (1996)

[GMR89] Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. SICOMP 18(1), 186–208 (1989)

[GMW91] Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J. ACM 38(3), 690–728 (1991)

[GO94] Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. Journal of Cryptology 7, 1–32 (1994)

[Gol02] Goldreich, O.: Concurrent zero-knowledge with timing, revisited. In: STOC 2002, pp. 332–340 (2002)

[HILL99] Håstad, J., Impagliazzo, R., Levin, L., Luby, M.: A pseudorandom generator from any one-way function. SICOMP 28, 12–24 (1999)

[Hol07] Holenstein, T.: Parallel repetition: simplifications and the no-signaling case. In: STOC 2007, pp. 411–419 (2007)

[HPPW08] Håstad, J., Pass, R., Pietrzak, K., Wikström, D.: An efficient parallel repetition theorem (2008) (manuscript)

[HRS09] Haitner, I., Rosen, A., Shaltiel, R.: On the (im)possibility of arthur-merlin witness hiding protocols. In: TCC 2009, pp. 220–237 (2009)

[IJK07] Impagliazzo, R., Jaiswal, R., Kabanets, V.: Chernoff-type direct product theorems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 500–516. Springer, Heidelberg (2007)

[IW97] Impagliazzo, R., Wigderson, A.: P = BPP if e requires exponential circuits: Derandomizing the xor lemma. In: STOC 1997, pp. 220–229 (1997)

[Kat08] Katz, J.: Which languages have 4-round zero-knowledge proofs? In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 73–88. Springer, Heidelberg (2008)

[KP01] Kilian, J., Petrank, E.: Concurrent and resettable zero-knowledge in poly-logarithmic rounds. In: STOC 2001, pp. 560–569 (2001)

[KPR98] Kilian, J., Petrank, E., Rackoff, C.: Lower bounds for zero knowledge on the internet. In: FOCS 1998, pp. 484–492 (1998)

[Lin03] Lindell, Y.: Bounded-concurrent secure two-party computation without setup assumptions. In: STOC 2003, pp. 683–692 (2003)

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

[PRS02] Prabhakaran, M., Rosen, A., Sahai, A.: Concurrent zero knowledge with logarithmic round-complexity. In: FOCS 2002, pp. 366–375 (2002)

[PV07] Pass, R., Venkitasubramaniam, M.: An efficient parallel repetition theorem for arthur-merlin games. In: STOC 2007, pp. 420–429 (2007)

[Raz98] Raz, R.: A parallel repetition theorem. SICOMP 27(3), 763–803 (1998)

[RK99] Richardson, R., Kilian, J.: On the concurrent composition of zero-knowledge proofs. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 415–432. Springer, Heidelberg (1999)

[Ros00] Rosen, A.: A note on the round-complexity of concurrent zero-knowledge. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 451–468. Springer, Heidelberg (2000)