SIAM J. COMPUT.
Vol. 40, No. 6, pp. 1529–1553

ON THE COMPOSITION OF PUBLIC-COIN ZERO-KNOWLEDGE PROTOCOLS*

RAFAEL PASS†, WEI-LUNG DUSTIN TSENG†, AND DOUGLAS WIKSTRÖM‡

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 setup) and in the bare public key model (where the prover and the verifier have registered public keys). We complement this result by constructing a public-coin black-box zero-knowledge proof based on one-way functions that remains secure under any a priori bounded number of concurrent executions. A key step (of independent interest) in the analysis of our lower bound shows that any public-coin protocol, when repeated sufficiently in parallel, satisfies a notion of "resettable soundness" if the verifier picks its random coins using a pseudorandom function.

Key words. public-coin interactive protocols, zero-knowledge, parallel repetition

AMS subject classification. 68Q17

DOI. 10.1137/100811465

1. Introduction.

Zero-knowledge (ZK) interactive protocols [22] 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. This is formalized by requiring that the view of an adversarial verifier, V^*, during an interaction with the prover, P, can be efficiently reconstructed by a so-called *simulator*, S. A particularly attractive notion of ZK, called *black-box* ZK [19], requires the existence of a universal simulator S that can generate the view of any V^* when given *black-box* access to V^*.

A fundamental question regarding ZK protocols is whether their composition remains ZK. Three basic notions of compositions are sequential composition [22, 19], parallel composition [15, 17], and concurrent composition [15, 13]. In a sequential composition, the players sequentially run many instances of a ZK 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 [19], this no longer holds for parallel composition [17] (and thus not for concurrent composition either). However, there are ZK protocols for all of NP that have been demonstrated to be secure under both parallel and concurrent composition. For the case of parallel composition, constant-round protocols are known [21, 15, 16]. For the case of concurrent composition, a series of work [37, 31, 35] show feasibility of $\tilde{O}(\log n)$-round black-box ZK protocols; furthermore, this round complexity is essentially optimal

*Received by the editors October 12, 2010; accepted for publication (in revised form) June 25, 2011; published electronically December 8, 2011. A preliminary version of this work appeared in CRYPTO '09.
http://www.siam.org/journals/sicomp/40-6/81146.html

†Department of Computer Science, Cornell University, Ithaca, NY 14853 (rafael@cs.cornell.edu, wdtseng@cs.cornell.edu). The first author's work was supported in part by a Microsoft New Faculty Fellowship, NSF CAREER award CCF-0746990, AFOSR award FA9550-08-1-0197, BSF grant 2006317, and I3P grant 2006CS-001-0000001-02. The second author's work was supported in part by an NSF Graduate Fellowship.

‡Royal Institute of Technology (KTH), SE-100 44 Stockholm, Sweden (dog@csc.kth.se).

with respect to black-box ZK [30, 38, 10].

Whereas the original ZK protocols of [22, 18, 8] 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 seminal paper, Goldreich and Krawczyk [17] show 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., [18, 8] 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 [17] 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?*

1.1. 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, regardless of round complexity.

THEOREM (informal). *If L has a public-coin argument that is black-box ZK and secure under parallel composition, then $L \in BPP$.*

In fact, our result establishes that any public-coin, black-box ZK protocol for a nontrivial language 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 ZK for NP that is secure under m-bounded concurrent composition.*

An earlier result of Barak [5] also constructs public-coin bounded-concurrent ZK protocols that additionally have *constant* rounds. However, Barak's construction is an *argument* (rather than a proof), assumes *collision-resistant hash function*, and uses *non-black-box* simulation.

Finally, we briefly turn to compositions in models with trusted setup. Canetti et al. [9] 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 earlier, $\tilde{\Omega}(\log n)$ rounds are necessary for nontrivial languages [10]). We show that for the case of public-coin protocols, the BPK setup 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 $L \in BPP$.*

We remark that our lower bound does not extend to more elaborate public key setups. For example, Damgård [12] shows that a public key infrastructure with a certification authority can be used to construct constant-round public-coin arguments that are black-box concurrent ZK.

As we will see, some of the intermediate ideas in our work are closely related to the notion of *resettable soundness* [2]. Very informally, we establish that parallel repetition of public-coin protocols not only reduces the soundness error [34, 25], but also *qualitatively* strengthens the soundness—roughly speaking, the new protocols will be secure under a "resetting" attack.

1.2. Techniques. To describe our techniques, first recall the Goldreich–Krawczyk [17] lower bound that only languages in BPP have $O(1)$-round public-coin black-box ZK protocols. Let $\Pi = \langle P, V\rangle$ be a public-coin black-box ZK protocol for a language L, and consider an adversarial verifier V^* that, instead of picking its messages at random, computes them by applying a hash function to the current transcript. It is shown in [17] that any black-box simulator S, together with V^*, can decide L: on input x, simply run $S^{V^*}(x)$ and accept if S outputs an accepting view of V^*. Using the ZK property of Π, if $x \in L$, then $S^{V^*}(x)$ will output an accepting view of V^* (because an honest prover would convince V^*). The crux of their proof is then to show that if $x \notin L$, then $S^{V^*}(x)$ will not output an accepting view. If S does not rewind V^*, then this would directly follow from the soundness of Π. However, S may rewind V^*, and may only convince V^* in one of its rewinding "threads." Nonetheless, [17] manages to show that if S, by rewinding or "resetting" V^*, manages to trick V^* into accepting $x \notin L$, then we can construct a machine T (based on S) that manages to convince an external verifier V (without rewinding V), contradicting the soundness of the protocol. In other words, they show that any $O(1)$-round public-coin protocol is sound under a resetting attack [9, 2], where the statement is fixed and the prover (simulator) running time is bounded by a fixed polynomial. Analogously, to prove our results, we show that any public-coin interactive protocol, repeated sufficiently many times in parallel (and again letting the verifier pick its messages by applying a hash function to the transcript), is sound under a resetting attack.

Previous reductions. The work of [17], as well as all subsequent black-box lower bounds (e.g., [30, 38, 10, 4, 29, 23]) relies on the following approach for constructing the stand-alone (nonresetting) prover T, given the rewinding simulator S. T incorporates S and internally emulates an execution of S with an internally emulated verifier (which of course can be rewound). During the emulation, T appropriately picks some messages sent by S to the internal verifier, and forwards them to an external verifier (and also forwards back the responses). The crux of the various lower bounds lies in choosing the externally forwarded messages so that the external verifier is convinced. 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 S will eventually choose this message to "continue" its simulation (and use it as part of the output view), or treat this message simply as a "rewinding" (used to collect information).

For the case of constant-round protocols, [17] shows that externally forwarding a *random* selection of messages works; if the protocol has d rounds, then this random selection is "correct" with probability at least $1/q^d$, where q is the number of queries made by the simulator to the verifier. This 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.[1]

Our reduction. In our work, we are given a ZK protocol $\Pi = \langle P, V\rangle$ for a language L that is secure under parallel repetitions. Building on the same framework

[1]For the case of sublogarithmic-round protocols, Canetti et al. [10] show that when given the freedom to construct a concurrent 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. Their work has the advantage that it applies to private-coin ZK protocols, but is not applicable in our setting due to the use of *concurrent* adversarial verifiers and being limited to *sublogarithmic-round protocols*. Incidentally, they also run the simulator S in a straight-line manner.

as [17], we let V^{m*} be a verifier that starts m parallel sessions and generates its messages using hash functions, let S be the black-box ZK simulator, and use $S^{V^{m*}}$ to decide L. As we will see, we choose the number of parallel sessions, m, as a (polynomial) function of the number of rounds in Π. Following the same argument, it is enough to show that on input $x \notin L$, S cannot produce an accepting view of V^{m*}. Because we may view S as a rewinding/resetting prover, it is equivalent to show that protocol $\langle P^m, V^{m*} \rangle$ is sound under resetting attacks. In the rest of this section we omit the common input x.

The crux of our work, then, is the following reduction: Given S, a *resetting* cheating prover of the *parallelized* protocol that convinces V^{m*}, we show how to construct T, a *straight-line* (nonrewinding) cheating prover of the original *single session* protocol that convinces V; this contradicts the soundness of protocol Π. To further clarify the difference between S and T, let us compare the transcripts of an interaction between T and V, and of an interaction between S and V^{m*}. A transcript of the interaction between T and V is simply a transcript of a *single session* of the protocol Π; each query from T to V is simply a prefix of the transcript that extends the previous query by one round of the protocol. A transcript of the interaction between S and V^{m*} can be much longer due to rewinds; furthermore, each query from S to V^{m*} is a prefix of a transcript of the *parallelized* protocol.

On a high level, T internally runs S with an internally simulated V^{m*}, and externally interacts with an external verifier V. In order to take advantage of S to convince the external verifier V, T "embeds" the interaction with V into the interaction between S and V^{m*}. This "embedding" is not straightforward for the following two reasons. First, just as in [17], the external verifier V cannot be reset, whereas S may reset V^{m*} many times (i.e., S can make many more queries than the number of rounds of the protocol); as we will explain shortly, T carefully picks a subset of the rewindings to forward externally. Second, recall that V is a single session verifier, whereas V^{m*} is an m-session parallel verifier (looking forward, the reason we let V be a single session verifier is to enable T to appropriately pick which rewindings to forward). Therefore, T embeds the interaction with V only into a *single session* i of the m parallel sessions in the interaction between S and V^{m*}; in fact, session i is picked uniformly random at the beginning and *fixed* throughout the execution of the reduction (looking forward again, the fact that session i is picked uniformly will be important for our analysis).

To summarize, T externally forwards only a subset of the S queries, and forwards only component i (corresponding to session i) of those queries. T then forwards back external responses from V as component i of the same subset of V^{m*} responses; all other V^{m*} responses are picked uniformly at random by T internally (this includes all except component i in the responses to the selected subset of S queries, and all components of the remaining responses). Here we rely on the fact that Π is public-coin in order for T to generate V^{m*} responses in the forwarded session, despite the fact that other verifier responses in the forwarded session may be externally generated by V.

Recall that the difficulty of the reduction comes from choosing which S queries to forward externally. As remarked earlier, the approach of running S in a straight-line manner seems unlikely to work for polynomial-round protocols. Instead, we let T *rewind* S (while S itself believes it is rewinding the internally simulated V^{m*}). Our strategy is twofold. First, T externally forwards only (component i of) queries that have a good chance of being included by S in its output (by assumption, S outputs a sequence of queries that convinces V^{m*}); because the protocol is public-coin, we can estimate this chance by doing internal test runs. Second, once we have forwarded (component i of) a query, we "force" S to include the query in its output by repeatedly rewinding S while repicking the internally generated V^{m*} messages (thus skewing the distribution of the internally generated V^{m*} messages).

To analyze T, we need to show that S would successfully convince the internally simulated V^{m*}, even though T has embedded the external interaction with V into the interaction between S and V^{m*}. Note that the success probability of S depends only on two inputs: the internally simulated V^{m*} messages, and the embedded external V messages (these can be found only in the forwarded session i). These two types of messages differ in that the internally simulated V^{m*} messages are picked by T, through the help of test runs, to be "good," while the external V messages are just uniform samples. We first show that if T is also allowed to rewind the external verifier V (which we cannot), ensuring that internal V^{m*} messages and external V responses are both "good," then T need only perform polynomially many rewinds in order for S to successfully convince V^{m*}. Next, to remove the assumption of rewinding V, we use a probabilistic lemma due to Raz [36], originally used to prove that parallel repetition reduces the soundness error in two-prover games. We show that if there are enough parallel sessions, then not being able to pick "good" verifier responses in just one *random* session introduces only a small statistical error; since session i is picked uniformly at random at the beginning, this suffices for bounding the success probability of T.

ZK lower bounds and soundness amplification. As an independent contribution, we believe that our techniques elucidate an intriguing (and useful) connection between *lower bounds* for black-box ZK and feasibility results for soundness/hardness amplification. Our techniques share many similarities with works on soundness amplification under parallel repetitions, such as [7, 34, 27], and especially [25]; in particular, our use of Raz's lemma is similar to its use in [25]. Whereas those works show how to transform a parallel prover with "small" success probability into a stand-alone prover with "high" success probability, we have adapted their techniques to transform a *rewinding/resetting* parallel prover into a *nonrewinding* stand-alone prover.

As a further example of this connection, we extend our lower bound to the BPK model by relying again on techniques developed for soundness amplification. In the BPK model, we have the additional problem that the external verifier can decide whether to accept or reject based on its secret key, which T does not know. Consequently, T cannot determine whether the external verifier would accept or reject when doing test runs, which is crucial for deciding which messages to forward externally. By relying on the "trust-halving" technique from [28, 7], and its refinement in [25], we show how T can make "educated guesses" on whether the external verifier accepts or not.

Extension to resettable soundness. More generally, the above techniques show how to transform a public-coin protocol so that it is sound under a weak form of resetting attack: where the statement is fixed, and the number of resets is a priori bounded. Simply take a public-coin protocol, sufficiently repeat it in parallel, and let the verifier generate its messages by applying hash functions to the current transcript. If the verifier uses pseudorandom functions instead of hash functions as in [2], then we may remove the a priori bound on the number of resets. Additionally, we show that if the original protocol is also a *proof of knowledge* [22, 15, 3], then the parallelized version satisfies the original (strongest) notion of resettable soundness from [2], where the adversarial prover can also change the statement between resets. A similar type of result for $O(1)$-round public-coin proofs of knowledge was shown in [2].

Outline. We give some preliminaries in section 2 and jump into our impossibility results in section 3 (standard model) and section 4 (bare public key model). We then present our public-coin bounded-concurrent ZK protocol in section 5. Details of our application to resettable soundness can be found in section 6.

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 $\{1,\ldots,n\}$.

2.1. Interactive protocols. An interactive protocol Π is a pair of interactive Turing machines, $\langle P,V\rangle$, 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 without loss of generality 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. 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 disjoint 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 aborts in all sessions), and accepts if and only if all m sessions are accepted by V.

2.2. ZK protocols. In the setting of ZK, 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 PPT 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 $\mathrm{View}^P_{V^*}(x,z)$ 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 an honest verifier is just the transcript of the interaction.

A *black-box simulator* S is a PPT machine that is given black-box access to V^* (written as $S=S^{V^*}$). Formally, S fixes the random coins r of V^* 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_r^* 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 share a common prefix.

Intuitively, an interactive proof is 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 ZK [22, 19]). *Let* $\Pi=\langle P,V\rangle$ *be an interactive proof (or argument) for a language* L. Π *is* black-box ZK *if there exists a black-box simulator* S *such that for every common input* x*, auxiliary input* z *and every (stand-alone) adversary* V^**,* $S^{V^*(x,z)}(x)$ *runs in time polynomial in* $|x|$*, and the following two ensembles* *are computationally indistinguishable as a function of* $|x|$*:*

$$\{\mathrm{View}_{V^*}^{P}(x,z)\}_{x\in L, z\in\{0,1\}^*} \approx \{S^{V^*(x,z)}(x)\}_{x\in L, z\in\{0,1\}^*}.$$

Note that because we consider black-box simulation, S does not get access to any "internals" of V^* such as its auxiliary input z.

DEFINITION 2 (black-box concurrent ZK [13]). *Let* $\Pi = \langle P, V\rangle$ *be an interactive proof (or argument) for a language* L. Π *is* black-box concurrent ZK *if for every polynomial* 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 following ensembles are computationally indistinguishable as a function of* $|x|$*:*

$$\{\mathrm{View}_{V^*}^{P}(x,z)\}_{x\in L, z\in\{0,1\}^*} \approx \{S_m^{V^*(x,z)}(x)\}_{x\in L, z\in\{0,1\}^*}.$$

We also consider a bounded version of concurrent ZK where the order of quantifiers are reversed [5].

DEFINITION 3 (black-box bounded concurrent ZK). *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 ZK *if there exists a black-box simulator* S *such that for every common input* x*, auxiliary input* z*, and every* m*-session concurrent adversary* V^**,* $S^{V^*(x,z)}(x)$ *runs in time polynomial in* $|x|$*. Furthermore, the following ensembles are computationally indistinguishable as a function of* $|x|$*:*

$$\{\mathrm{View}_{V^*}^{P}(x,z)\}_{x\in L, z\in\{0,1\}^*} \approx \{S^{V^*(x,z)}(x)\}_{x\in L, z\in\{0,1\}^*}.$$

2.3. Resettable soundness. Informally, given a protocol $\Pi = \langle P, V\rangle$, a cheating prover P^* performing a *resetting attack* has the power to reset (i.e., rewind) the honest *resettable* verifier, resulting in multiple *sessions* of Π. Furthermore, in all these sessions, V uses the same random tape that is uniformly chosen before the attack. For example, a black-box ZK simulator is a valid resetting attack. We can consider two different models on how the input instances are chosen for each session. In the model of *resettable soundness* as defined by [2], P^* can adaptively choose different input instances for each session. We also consider the model where P^* is given an input instance that must be used in all sessions (similar to the definition of resettable ZK by [9]); we call this *fixed-input resettable soundness*.

DEFINITION 4 (resetting attack [2, Definition 3.1]). *A* resetting attack *of a cheating prover* P^* *on a* resettable verifier V *is defined by the following two-step random process, indexed by a security parameter* n*:*

1. *Uniformly select and fix* $t = \mathrm{poly}(n)$ *random tapes, denoted* $r_1, \ldots, r_t$*, for* V*, resulting in deterministic strategies* V_{r_j}*. When an input* $x \in \{0,1\}^n$ *is also chosen, we call* $V_{r_j}(x)$ *an* incarnation *of* V *(i.e.,* V *with its randomness set to* r_j *and common input set fixed to* x*).*
2. *On input* 1^n*,* P^* *is allowed to interact with* $\mathrm{poly}(n)$ *incarnations of* V*.* P^* *chooses each incarnation (adaptively) by choosing* $x \in \{0,1\}^n$ *and* $j \in [t]$ *(these choices may depend on* P^**'s previous interactions with other incarnations of* V*).* P^* *may freely switch among interactions with different incarnations of* V*, and may rewind/reset each incarnation of* V*.*

We further define two variants of resetting attacks. In a fixed-input resetting attack*, the cheating prover* P^* *is given a fixed input instance* x *to use in all sessions. In a* q-query resetting attack*, the cheating prover* P^* *is allowed* q *queries total for verifier messages (summed over all interactions among the different incarnations of* V*).*

Remark. We have chosen the "interleaving" attack model instead of the "noninterleaving" attack model, where P^* must finish its current interaction with an incarnation of V completely, before starting another interaction (see discussions in [9, 2]). The two models are equivalent as shown in [9]. We choose the "interleaving" model because later we will make the assumption that P^* never makes the same query to V twice. The notion of a q-query resetting attack is also more natural in the "interleaving" model.

DEFINITION 5 (resettable soundness [2, Definition 3.1]). *Let $\Pi = \langle P, V\rangle$ be a pair of interactive machines, where V is PPT. We say that Π is a* resettably sound proof *(resp.,* resettably sound argument*) for a language L if the following condition holds:*

> Resettable soundness: *For every resetting attack by P^* (resp., polynomial-size P^*), the probability that some incarnation $V_r(x)$ accepts and $x \notin L$ is negligible in n.*

*We say Π is a q-*query fixed-input resettably sound proof *(resp.,* argument*) for a language L if the resettable soundness property holds with respect to any q-query fixed-input resetting attack.*

3. Impossibility of public-coin black-box parallel ZK. In this section we show that only languages in BPP have public-coin concurrent ZP protocols. We actually show a stronger result: Except for languages in BPP, no public-coin protocol remains black-box ZK when repeated in parallel. The formal theorems are stated below, where n denotes the security parameter or the input size.

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

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

The difference between Theorems 1 and 2 is caused by the difference between proofs and arguments. While the two theorems differ slightly in parameters, their proofs differ greatly. We remark that our theorems trivially hold with respect to "nonaborting" verifiers since we focus only on public-coin protocols.

3.1. Reducing to resettable soundness. The proofs of Theorems 1 and 2 begin in the same high-level framework as that of [17]. Suppose that a language L has a k-round, public-coin ZK protocol $\Pi = \langle P, V\rangle$, and that Π^m is ZK with a black-box simulator S that runs in time n^d. To show that $L \in$ BPP, we construct a "random looking" adversarial verifier, V^*, and consider the following decision algorithm D: $D(x)$ runs $S^{V^*}(x)$ 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 [17, 11]). Given $h \leftarrow H$, let V_h^* be the verifier that when queried with transcript τ, responds (deterministically) with the message $h(\tau)$. We write $V^* = V_H^*$ to mean V_h^* for a randomly chosen h; i.e., when D runs $S^{V_H^*}$, D first chooses h randomly from H and then runs $S^{V_h^*}$.

We make two easy observations about S^{V^*} due to [17]. 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_H^* 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 an interaction of $\langle P^m, V^* \rangle$ since S is a ZK 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 (made possible since V is public-coin). 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. Equivalently, we may view S as an n^d-query fixed-input resettable prover, and we show that the protocol $\langle P^m, V^* \rangle$ is n^d-query fixed-input resettable sound. Therefore, Theorems 1 and 2 are completed by the following lemmas, respectively.

LEMMA 3 (resettably sound proofs). *Suppose that* $\Pi = \langle P, V \rangle$ *is a* $k = \mathrm{poly}(n)$*-round public-coin black-box ZK proof with soundness error* $1/2$*. If* $m \geq k \log^2 n$ *and* H *is a family of* $q = \mathrm{poly}(n)$*-wise independent hash functions, then* $\langle P^m, V_H^* \rangle$ *is* q*-query fixed-input resettably sound.*

LEMMA 4 (resettably sound arguments). *Suppose that* $\Pi = \langle P, V \rangle$ *is a* $k = \mathrm{poly}(n)$*-round public-coin black-box ZK argument with soundness error* $1/2$*. If* $m \geq k^2 \log^2 n$ *and* H *is a family of* $q = \mathrm{poly}(n)$*-wise independent hash functions, then* $\langle P^m, V_H^* \rangle$ *is* q*-query fixed-input resettably sound.*

Remark. Lemmas 3 and 4 may be stronger than necessary in two ways. First, the definition of resettable soundness requires negligible soundness error while our main theorems require only soundness error 1/3. Second, the definition of resettable soundness allows the resetting prover to interact with polynomially many copies of V_h^* with uniformly and independently chosen h's, while the ZK simulator interacts only with one copy of V_h^* for a uniformly chosen h. This second difference is moot, however, because it is trivial to reduce a resetting attack on polynomially many copies of V_h^* (with uniformly and independently chosen h's) to a resetting attack on a single copy of V_h^* (with uniformly chosen h), with only a polynomial loss in success probability. Therefore, in our proofs for Lemmas 3 and 4, we consider only one copy of V_h^*.

3.2. Proof of Lemma 3: Resettably sound proofs. Using the soundness amplification theorem of [1], protocol $\langle P^m, V_H^* \rangle$ has soundness error at most $1/2^m$. Let $\hat{P}^*$ be a q-query fixed-input resettable prover. Suppose for the sake of contradiction that for some input $x \notin L$, V_H^* accepts a resettable interaction with $\hat{P}^*$ with probability $1/p(n)$ for some polynomial p. We follow the strategy of [17] to use $\hat{P}^*$ in order to break the soundness of $\langle P^m, V_H^* \rangle$.

Whenever $\hat{P}^*$ succeeds in breaking resettable soundness, $\hat{P}^*$ would have queried V^* for k verifier messages that together form an accepting transcript of Π^m. A cheating prover of Π^m can therefore run $\hat{P}^*$ internally, guess which queries of $\hat{P}^*$ will form the accepting transcript, and forward them to an outside honest verifier of Π^m. Since $\hat{P}^*$ queries V^* for at most $q(n)$ messages, the probability of guessing all the right queries is at least q^{-k} (one guess for each round of Π). Note that forwarding queries to an outside honest verifier does not lower the success probability of $\hat{P}^*$ since V^* is identical to an honest verifier (they both respond with random messages). Thus this cheating prover, using $\hat{P}^*$, can break the soundness of Π^m with probability at least

$(1/p)q^{-k} = 2^{-\Theta(k \log n)}$. Since $m \geq k \log^2 n$, we have $2^{-m} < 2^{-\Theta(k \log n)}$ and reach a contradiction. $\square$

3.3. Proof of Lemma 4: Resettably sound arguments. We turn to proving our main result. Again we argue by contradiction. Suppose that $\hat{P}^*$ is a q-query fixed-input resettable prover, and suppose that $\hat{P}^*$ convinces V_H^* on some input $x \notin L$ with probability more than $1/p(n)$ for some polynomial p. We cannot repeat the proof of Lemma 3 because parallel repetitions cannot reduce the soundness of arguments beyond being negligibly small. Instead, we directly show a parallel repetition theorem for resettable soundness; that is, we relate the resettable soundness of $\langle P^m, V_H^* \rangle$ to the soundness of Π.

Proof outline. The rest of this section describes how to construct a cheating prover T for Π. T runs $\hat{P}^*$ internally and simulates V_H^* in response to $\hat{P}^*$ queries. Every query made by $\hat{P}^*$ is answered by a uniformly random reply. This perfectly simulates V_H^* since H is q-wise independent and $\hat{P}^*$ makes at most q queries (and never makes the same query twice); at the end of the qth query, T will have implicitly defined a hash function $h \in H$ and simulated V_h^*, and $\hat{P}^*$ will have successfully broken resettable soundness with probability $1/p(n)$ over the choice of these random replies (i.e., generated an accepting view of V_H^*).

To break the (stand-alone) soundness of Π, T chooses one of the m parallel sessions and forwards a complete set of $\hat{P}^*$ queries in that session (one for each round of Π) to an honest outside verifier V. The goal is to forward the queries on which $\hat{P}^*$ is able to convince $V^* = V_H^*$ in protocol Π^m. This is challenging because $\hat{P}^*$ may have multiple queries for each round of Π^m. While T must decide to forward a query or not at the time of the query, $\hat{P}^*$ can wait until all queries are completed before choosing which queries to form an accepting view of V^*. To overcome this obstacle, a key part of our analysis relies on *rewinding* $\hat{P}^*$ (note that at the same time, $\hat{P}^*$ believes that it is rewinding V^*). Our strategy is twofold. First we forward only queries that have some chance (preferably a good chance) of being included in a convincing transcript; this is done by doing test runs of $\hat{P}^*$. Once we have forwarded a query, we force $\hat{P}^*$ to use the query to convince V^*, by repeatedly rewinding $\hat{P}^*$.

We describe a *transcript* of $\hat{P}^*$ as an alternating sequence of responses from T and queries from $\hat{P}^*$, $[t_1, s_1, t_2, s_2, \ldots]$, where each $\hat{P}^*$-query s_i 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^* (transcripts of Π^m), while h and $[\cdot]$ denote transcripts of $\hat{P}^*$ (transcripts of a resettable execution of Π^m). The goal of T is then to generate a full transcript h of $\hat{P}^*$ in which $\hat{P}^*$ generates a convincing transcript τ of $\langle P^m, V^* \rangle$, while simultaneously having the foresight to forward (a session of) all the $\hat{P}^*$-queries pertaining to τ to the external verifier V (i.e., all $\hat{P}^*$-queries in h that are a prefix of τ). If so, T has broken the soundness of Π, and we call this a *successful* simulation of $\hat{P}^*$. Note that because the randomness of $\hat{P}^*$ is fixed, the behavior of $\hat{P}^*$ is entirely determined by the T-responses in a transcript.

We start with a brief description of T. T first fixes a random session $\tilde{\jmath} \in \{1, \ldots, m\}$ to be forwarded. Then in k iterations (one for each round of Π), T incrementally fixes a transcript of $\hat{P}^*$ and forwards a $\hat{P}^*$-query to V. In more detail, at the beginning of iteration i, T starts with a partial transcript $h_i = [t_1, s_1, \ldots, s_\ell]$ of $\hat{P}^*$ that ends with $s_\ell = \tau_i$, a query for the ith message of Π ($h_1 = []$, the empty transcript). Then,

Step 1. T forwards session $\tilde{\jmath}$ of the query τ_i to V, and receives a response $v_i^{(\tilde{\jmath})}$;

Step 2. Fixing the reply $v_i^{(\tilde{\jmath})}$, T uniformly samples completions of the partial transcript h_i until a "successful" completion h is found; specifically, $\hat{P}^*$ on tran-

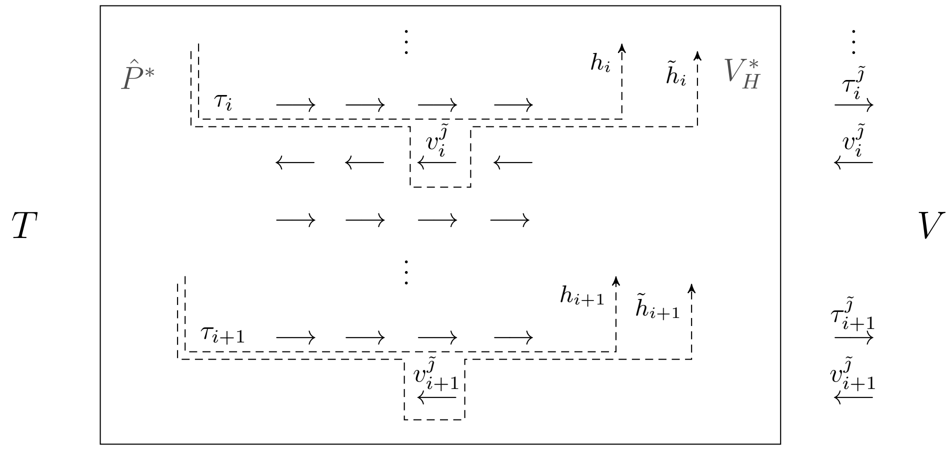

FIG. 1. *In order to interact with an outside honest verifier V, the reduction T internally maintains a partial interaction between the given resetting prover, $\hat{P}^*$, and the (supposedly resettably sound) verifier V_H^*. The figure captures T after step 1 of the $i+1$st iteration and illustrates some of the notation we define in the analysis.*

script h should produce an accepting view of V^*, τ, that extends the query τ_i. To move onto the next iteration, let τ_{i+1} be the length $i+1$ prefix of τ, and let h_{i+1} be the prefix of h up until $\hat{P}^*$ makes the query τ_{i+1}.

During the analysis, we first use Raz's lemma to show that because the number of sessions is large and $\tilde{\jmath}$ was chosen randomly, we may pretend that $v_i^{(\tilde{\jmath})}$ 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 T; all our hybrids generate truly random responses to $\hat{P}^*$-queries so that $\hat{P}^*$ cannot distinguish the hybrids from V^*. We start with a hypothetical hybrid and gradually move towards T. Refer to Figure 1 for a graphical description of the reduction T.

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

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

We now describe $T^{(1)}$ in detail. In the very beginning, $T^{(1)}$ fixes a random session $\tilde{\jmath} \in \{1, \ldots, m\}$; eventually the $\tilde{\jmath}$th session will be forwarded externally. After that, $T^{(1)}$ incrementally grows a transcript of $\hat{P}^*$ in k iterations. During the ith iteration, $T^{(1)}$ receives a partial transcript of $\hat{P}^*$ from the previous iteration, $h_i = [t_1, s_1, \ldots, s_\ell = \tau_i]$, where τ_i is a $\hat{P}^*$-query for the ith verifier message of Π^m ($h_1 = []$, the empty transcript). As an invariant maintained by $T^{(1)}$, it should be possible to extend h_i into a full transcript of $\hat{P}^*$, where $\hat{P}^*$ 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 as follows. $T^{(1)}$ randomly samples a completion of h_i into h, conditioned on success (always possible due to the invariant). Let $v_i^{(\tilde{\jmath})}$ be the response to τ_i in the $\tilde{\jmath}$th session in the successful completion h. Let $\tilde{h}_i$ be a partial extension of the partial transcript h_i, where the session $\tilde{\jmath}$ response to τ_i is fixed to $v_i^{(\tilde{\jmath})}$ (but the responses in other sessions are not specified).

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 transcript $\tilde{h}$, $\hat{P}^*$ would output an accepting view τ of V^* (note that τ must extend τ_i). Let τ_{i+1} be the $\hat{P}^*$ query for the $i+1$st verifier message in τ (note that τ_{i+1} extends τ_i by definition of success). $T^{(1)}$ then sets h_{i+1} to be the prefix of $\tilde{h}$ up to when $\hat{P}^*$ makes the query τ_{i+1}. 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 (kth) iteration, $T^{(1)}$ simply outputs $\tilde{h}$ as a full transcript of $\hat{P}^*$ (there is no τ_{k+1} to fix). Due to the invariant, $T^{(1)}$ always produces a transcript of $\hat{P}^*$, on which $\hat{P}^*$ outputs an accepting transcript τ. Moreover, the prefixes of τ would be the same $\tau_1, \ldots, \tau_k$ that were "chosen" by $T^{(1)}$ in each iteration (and would eventually be forwarded to the external verifier V in later hybrids).

Hybrid 2. Our second hybrid, $T^{(2)}$, describes a way to *efficiently* 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 $100k^2pq$ times, until a successful completion is found. If none of the completions are successful, then $T^{(2)}$ 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 random completions that are needed to sample a successful one. In the following analysis we distinguish between two probability spaces: $\Pr_P[\cdot]$ is used to measure probabilities over a single execution of $\hat{P}^*$. 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 $\hat{P}^*$ multiple times.

Let H_i and $\tilde{H}_i$ be the set of possible partial transcripts of $\hat{P}^*$ that is given to $T^{(2)}$ in steps 1 and 2 of the ith iteration, respectively. Given $h \in H_i$ (or $\tilde{H}_i$), let $\Pr_P[h]$ denote the probability that a transcript of $\hat{P}^*$ has prefix h, and let $\Pr_T[h]$ denote the probability that $T^{(2)}$ is given h in the ith iteration; similarly, $\Pr_P[\cdot \mid h]$ and $\Pr_T[\cdot \mid h]$ are probabilities conditioned on these events occurring. Let A^h be the event (over the $\hat{P}^*$ probability space) that a transcript of $\hat{P}^*$ has prefix h and is a successful completion of h; as a special case, $A = A^{\emptyset}$ is just the event that $\hat{P}^*$ 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.

First we give a lemma. Intuitively, the lemma says that the probability of $T^{(2)}$ fixing h is proportional to the probability of successfully completing h; the normalizing factor is simply $\Pr_P[A]$, the probability that $\hat{P}^*$ produces an accepting transcript.

LEMMA 5. *Let* $h \in \tilde{H}_i$. $\Pr_T[h]\Pr_P[A] = \Pr_P[A^h]$.

Proof. Recall that the behavior of $\hat{P}^*$ is entirely determined by the random messages generated by $T^{(2)}$. Let us consider a complete binary tree $\mathcal{T}$ of depth n^d that represents all possible length n^d random bit strings generated by $T^{(2)}$. Then every partial execution of $\hat{P}^*$ corresponds to a node in $\mathcal{T}$ based on the verifier messages received so far by $\hat{P}^*$ in h.

Let us focus on the leaf nodes in $\mathcal{T}$ since they occur with equal probability. Given h, define $L(h)$ to be the set of leaf nodes in $\mathcal{T}$ that are children of h; these nodes correspond to possible completions of h. We also define $G(h)$ to be the subset of $L(h)$ that corresponds to successful completions of h (i.e., leaves where the event A^h is true). Finally, let $L_0 = L(\emptyset)$ be all the leaf nodes, and let $G_0 = G(\emptyset)$ be the subset of L_0 that corresponds to executions where $\hat{P}^*$ produces an accepting transcript.

Recall that our goal is to prove that

$$\Pr_T[h] \Pr_P[A] = \Pr_P[A^h].$$

Clearly

$$\Pr_P[A] = \frac{|G_0|}{|L_0|} \quad \Pr_P[A^h] = \frac{|G(h)|}{|L_0|}. \tag{1}$$

To expand $\Pr_T[h]$, let $\tilde{h}_1, h_2, \ldots, h_i, \tilde{h}_i = h$ be the prefixes of h given to $T^{(1)}$ in previous steps of previous iterations. As we see below, the expression for $\Pr_T[h]$ telescopes

$$\begin{aligned}
\Pr_T[h] &= \Pr_T[\tilde{h}_1] \prod_{\ell=2}^{i} \Pr_T[h_\ell \mid \tilde{h}_{\ell-1}] \Pr_T[\tilde{h}_\ell \mid h_\ell] \\
&= \frac{|G(\tilde{h}_1)|}{|G_0|} \prod_{\ell=2}^{i} \frac{|G(h_\ell)|}{|G(\tilde{h}_{\ell-1})|} \frac{|G(\tilde{h}_\ell)|}{|G(h_\ell)|} \\
&= \frac{|G(\tilde{h}_i)|}{|G_0|} = \frac{|G(h)|}{|G_0|}.
\end{aligned} \tag{2}$$

Equations (1) and (2) together give the lemma. ∎

Now we bound the expected number of samples needed to find a successful completion.

LEMMA 6. $\mathbb{E}_T[R_i] \leq pq$.

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

$$\mathop{\mathbb{E}}_T[R_i] = \sum_{h \in \tilde{H}_i} \Pr_T[h] \mathop{\mathbb{E}}_T[R_i \mid h]. \tag{3}$$

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

$$\mathop{\mathbb{E}}_T[R_i \mid h] = \frac{1}{\Pr_P[A^h \mid h]} \;\Rightarrow\; \mathop{\mathbb{E}}_T[R_i] = \sum_{h \in \tilde{H}_i} \Pr_T[h] \frac{1}{\Pr_P[A^h \mid h]}. \tag{4}$$

By expanding the right-hand side of Lemma 5 and rearranging terms, we have

$$\begin{aligned}
&\Pr_T[h] \Pr_P[A] = \Pr_P[A^h] = \Pr_P[h] \Pr_P[A^h \mid h] \\
\Rightarrow\ &\Pr_T[h] \frac{1}{\Pr_P[A^h \mid h]} = \Pr_P[h] \frac{1}{\Pr_P[A]} \leq p \Pr_P[h],
\end{aligned}$$

since we assumed $\Pr_P[A] \geq 1/p$. Substituting this back into (4) gives

$$\mathop{\mathbb{E}}_{T}[R_i] \leq p \sum_{h \in \tilde{H}_i} \Pr_P[h]. \tag{5}$$

Finally, we may break up the set $\tilde{H}_i$ based on the length of h which ranges from 1 to q (where length is the number of $\hat{P}^*$-queries). Since each transcript of $\hat{P}^*$ has exactly one length ℓ prefix,

$$\mathop{\mathbb{E}}_{T}[R_i] \;\leq\; p \sum_{\ell=1}^{q} \sum_{h \in \tilde{H}_i, |h|=\ell} \Pr_P[h] \;\leq\; p \sum_{\ell=1}^{q} 1 = pq. \qquad \square$$

Finally, we show that $100k^2pq$ random completions are enough for $T^{(2)}$.

LEMMA 7. *$T^{(2)}$ aborts with probability at most $1/5$.*

Proof. Since $\mathbb{E}_T[R_i] = \sum_{\tilde{h}_i} \Pr_T[\tilde{h}_i] \, \mathbb{E}_T[R_i \mid \tilde{h}_i] = \mathbb{E}_T[\mathbb{E}_T[R_i \mid \tilde{h}_i]] \leq pq$, the Markov inequality states that the probability of $T^{(2)}$ fixing an $\tilde{h}_i$ such that $\mathbb{E}_T[R_i \mid \tilde{h}_i] \geq 10kpq$ is at most $1/(10k)$. For each "good" $\tilde{h}_i$, where $\mathbb{E}_T[R_i \mid \tilde{h}_i] < 10kpq$, we apply the Markov inequality again to obtain $\Pr_T[R_i \geq 100k^2pq \mid \tilde{h}_i] \leq 1/(10k)$. 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 $1/5$. $\square$

Hybrid 3. Our third and final hybrid $T^{(3)} = T$ differs from $T^{(2)}$ in step 1 of each iteration. Recall that some session $\tilde{\jmath}$ is chosen randomly as the forwarding session. Instead of generating $v_i^{(\tilde{\jmath})}$ 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 $\hat{P}^*$ even if session $\tilde{\jmath}$ 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 $\tilde{\jmath}$ of τ_i to V and uses the response from V as $v_i^{(\tilde{\jmath})}$ in step 2.[2] Suppose for now that $T^{(3)}$ does not abort and terminates successfully. Then $\hat{P}^*$ would have generated an accepting transcript τ of Π^m. Since $\tau_1, \ldots, \tau_k$ are prefixes of τ, session $\tilde{\jmath}$ 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 [36, Claim 5.1] in analogy with [27, 25] to show that $v_i^{(\tilde{\jmath})}$ 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 [26, Lemma 5].

LEMMA 8. *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 let $\Pr[W]$ be measured according to the joint probability distribution $\Pi_j P_{U_j}$. Then*

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

[2]Strictly speaking, the interaction between $T^{(3)}$ and the honest verifier V is nonresetting. Therefore, instead of forwarding session $\tilde{\jmath}$ of query τ_i to V, $T^{(3)}$ simply sends the last prover message in session $\tilde{\jmath}$ of the query τ_i to V. For ease of exposition, we continue to use the phrase "$T^{(3)}$ forwards the query τ_i" to mean the above.

where Δ is the statistical distance between distributions, and $U_j|W$ is the jth 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 8 allows us to bound the change in success probability when $T^{(3)}$ forwards messages from a random session to V.

LEMMA 9. *$T^{(3)}$ 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).[3] 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 a successful completion of h. By definition, the distribution of $v_i^{(\tilde{\jmath})}$ produced by T_{i+1} (i.e., $T^{(2)}$) is just the first message of $U_{\tilde{\jmath}}|W^h$. On the other hand, the distribution of $v_i^{(\tilde{\jmath})}$ 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 differ only in how $v_i^{(\tilde{\jmath})}$ is produced, their difference in success probability can be bounded by the statistical difference in the distributions of $v_i^{(\tilde{\jmath})}$. This is in turn bounded by

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

Lemma 8 states that for any event W,

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

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 6 along with the Markov inequality implies that except with probability $1/(10k)$, $T^{(2)}$ fixes a "good" h with $\mathbb{E}_T[R_i \mid h] \leq 10kpq$, so that

$$\Pr[W^h] = \Pr_P[A^h \mid h] = \frac{1}{\mathbb{E}_T[R_i \mid h]} \geq \frac{1}{10kpq}.$$

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

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

[3] 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.

since statistical distances are upper bounded by 1, and

$$\sum_{\text{good } h \in H_i} \Pr_T[h] \left(\frac{1}{m} \sum_{j=1}^{m} \Delta(U_j | W^h, U_j) \right)$$
$$\leq \sum_{\text{good } h \in H_i} \Pr_T[h] \sqrt{\frac{1}{m} \log(10kpq)} \;\leq\; \sqrt{\frac{1}{m} \log(10kpq)},$$

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

$$\frac{1}{10k} + \sqrt{\frac{1}{m} \log\left(10kpq\right)} = \frac{1}{10k} + O\left(\frac{1}{k\sqrt{\log n}}\right),$$

since $m \geq k^2 \log^2 n$. Summing up over the hybrids, and recalling that $T^{(2)}$ fails with probability at most 1/5 (Lemma 7), $T^{(3)}$ fails with probability at most

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

as desired. ▯

Lemma 9 shows that T is successful with probability $> 1/2$, and completes the proof of Lemma 4. ▯

Remark. As with most lower bounds for black-box ZK, a careful reading reveals that Theorems 1 and 2 also apply to more liberal definitions of ZK, such as ε-ZK[4] [13] and ZK with expected polynomial time simulators.

4. Public-coin ZK in the BPK model. Many setup assumptions have been used to construct concurrent ZK with better efficiency than the standard model. For example, in the common reference string (CRS) model, even noninteractive ZK is possible [14]. Other "weaker" setups have produced varying results, and we will be concentrating on the *bare public key* model.

In the BPK model [9], every player has a public key that can be accessed by any other player. When a protocol is repeated in parallel, we assume that the honest parties use fresh independent public keys for each parallel session. By assuming that all public keys are properly registered before a protocol begins, Canetti et al. [9] showed that *constant-round*, private-coin arguments exist for NP even if we require black-box *resettable ZK*, a property that implies black-box concurrent ZK. In constrast, in the plain model, $\tilde{O}(\log n)$ rounds are required for concurrent black-box ZK proofs [10]. It is therefore natural to ask if the BPK setup can overcome our lowerbound for public-coin ZK protocols.

In this section we extend our impossibility result from section 3 to the BPK model. We actually extend our result to a larger class of *slightly private-coin* protocols, defined by the following properties:

1. The first message of the protocol, from the verifier, is allowed to be private coin. All other subsequence verifier messages are public-coin, i.e., independent segments of the verifier's random tape.
2. At the end of the protocol, the verifier may run a private coin algorithm to accept or reject the interaction. In particular, the verifier's decision may depend on the private coins used to generate the first message.

[4] In ε-ZK, the indistinguishability gap between the view of V^* and the view generated by the simulator is allowed to be an inverse polynomial, as opposed to negligible.

Note that every public-coin protocol in the BPK model can be transformed into a slightly private-coin protocol, because of the following:

1. The verifier can send its public key to the prover in the first message (property 1).
2. The verifier can base its acceptance decision on its secret key (property 2).

Our modified theorem is the following.

THEOREM 10. *Suppose that language L has a $k = \text{poly}(n)$-round slightly private-coin black-box ZK argument Π with negligible soundness error in n. If $m \geq (k^2 \log^2 k) \log^2 n$ and Π^m is ZK, then $L \in BPP$.*

Recall that in the analysis of Theorem 2, we treat the black-box ZK simulator S as a resetting prover $\hat{P}^*$ of $\langle P^m, V^* \rangle$, and we use $\hat{P}^*$ to construct a machine T, which in turn contradicts the soundness of Π. We now have a problem whenever T needs to sample a successful completion of a partial transcript of $\hat{P}^*$, since T does not know whether the external verifier V would accept or reject the transcript produced by $\hat{P}^*$. To overcome this problem, we follow an approach similar to [7, 25] by guessing whether V would accept or reject based on whether the other verifiers, simulated by T, accept or reject their respective parallel sessions.

Proof. We extend the analysis of Theorem 2 in analogy with [25]. We first describe how T guesses if V accepts or rejects in the forwarded session $\tilde{\jmath}$. Whenever T completes a partial execution of $\hat{P}^*$, let $z_{-\tilde{\jmath}}$ be the number of sessions, excluding session $\tilde{\jmath}$, in which S produced a rejecting view. We exclude session $\tilde{\jmath}$ for the aforementioned reason that without knowing the private key (or private coins) of the external verifier V, T cannot tell if V will accept or reject the view.

Let $w_{-\tilde{\jmath}}$ be a Bernoulli random variable with $\Pr[w_{-\tilde{\jmath}} = 1] = 2^{-\nu z_{-\tilde{\jmath}}}$, where ν is an asymptotically small parameter to be determined later. $w_{-\tilde{\jmath}}$ corresponds to T's guess: If $w_{-\tilde{\jmath}} = 1$, then T will consider the completion successful, and vice versa. Intuitively, T is more likely to consider a completion a success if the number of rejecting sessions is fewer.

To facilitate the analysis, we also consider a hypothetical but more symmetric process. Given a transcript generated by $\hat{P}^*$, let z be the number of sessions, including session $\tilde{\jmath}$, in which $\hat{P}^*$ produced a rejecting view. Similarly, let w be the Bernoulli random variable with $\Pr[w = 1] = 2^{-\nu z}$.

We now prove Theorem 10 with the same framework as Theorem 2, using the following modified hybrids. Hybrids $T^{(1)}$, $T^{(2)}$, and $T^{(3)}$ are constructed as before, except they now compute z and w to determine whether a completion is successful. The final machine, T, differs from $T^{(3)}$ by computing $z_{-\tilde{\jmath}}$ and $w_{-\tilde{\jmath}}$ instead.

LEMMA 11. *The probability that $T^{(1)}$ generates a rejecting view in session $\tilde{\jmath}$ is at most*

$$\frac{3}{m}\left(\frac{-\log \nu^2}{\nu} + 4\right).$$

Proof. The proof of this lemma essentially follows from an analysis in [25] (which contained more general parameters). For the sake of completeness, we include their analysis without the extra parameters here.

Before introducing the public key extension, $T^{(1)}$ simply samples a random successful transcript of $\hat{P}^*$ (see Lemma 5). After adopting the new notion of success based on w, $T^{(1)}$ now samples a random transcript of $\hat{P}^*$ conditioned on $w = 1$. That is, $T^{(1)}$ outputs a transcript of $\hat{P}^*$ that generates rejecting views in j sessions with probability proportional to $2^{-\nu j}$.

Since $T^{(1)}$ chooses $\tilde{\jmath}$ randomly, it is enough to bound the expected number of rejecting sessions. Let p_j be the probability that in a random execution of $\hat{P}^*$, the output view contains j rejecting sessions. Then, the expected number of rejecting verifiers is

$$\frac{\sum_{j=0}^{m} j p_j 2^{-\nu j}}{\sum_{j=0}^{m} p_j 2^{-\nu j}}. \tag{6}$$

A bound of (6) with more general parameters is given in [25]. For the sake of completeness, we include their analysis below without the extra parameters.

Recall that by assumption, $\hat{P}^*$ generates an output view in which all sessions accept with probability at least $1/3$. Therefore we can lower bound the denominator of (6) by

$$\sum_{j=0}^{m} p_j 2^{-\nu j} \geq p_0 \geq 1/3.$$

To upper bound the numerator, we use the following inequality:

$$\sum_{j=0}^{\infty} j 2^{-\nu j} = \frac{2^{-\nu}}{(1-2^{-\nu})^2} \leq \frac{1}{(1-2^{-\nu})^2} \leq \frac{4}{\nu^2}.$$

The last inequality follows from the fact that $1 - 2^{-\nu} \geq \nu/2$ for small ν. Directly applying this bound to the numerator (using $p_j \leq 1$) gives an overly loose bound since ν is asymptotically small. Instead, we split the expression of the numerator at some parameter t:

$$\begin{aligned}\sum_{j=0}^{m} j p_j 2^{-\nu j} &\leq t \sum_{j=0}^{m} p_j 2^{-\nu j} + \sum_{j=1}^{m-t} j p_{t+j} 2^{-\nu(t+j)} \\ &\leq t + \frac{4}{\nu^2} 2^{-\nu t}.\end{aligned}$$

Setting $t = -\log \nu^2/\nu$, we see that the expected number of rejecting verifiers is at most

$$3\left(\frac{-\log \nu^2}{\nu} + 4\right).$$

Since $T^{(1)}$ chooses $\tilde{\jmath}$ uniformly from $\{1, \ldots, k\}$, the probability that $T^{(1)}$ outputs a view that rejects in session $\tilde{\jmath}$ is

$$\frac{3}{m}\left(\frac{-\log \nu^2}{\nu} + 4\right). \qquad \square$$

LEMMA 12. *The probability that $T^{(2)}$ aborts is at most $1/5$. Otherwise, the output of $T^{(2)}$ is identical to $T^{(1)}$.*

Proof. By computing w and z, there are now more "successful" executions than before (originally, only executions where $z = 0$, i.e., no rejecting sessions, were successful). Therefore, $T^{(2)}$ now aborts with less probability than before, which is bounded by $1/5$ (Lemma 7). $\square$

LEMMA 13. *$T^{(3)}$ fails to produce an accepting view in session $\tilde{\jmath}$ with probability at most*

$$\frac{3}{m}\left(\frac{-\log \nu^2}{\nu}+4\right)+\frac{3}{10}+O\left(\frac{1}{\log n}\right).$$

Proof. This follows from Lemma 11, and by applying Raz's lemma in the same manner as in Lemma 9. ☐

LEMMA 14. *The output of $T^{(3)}$ and T differs statistically by at most $k\nu$.*

Proof. $T^{(3)}$ and T differs in how a successful completion is recognized. For any completion, the difference in probability of it being considered successful by $T^{(3)}$ and T is

$$\Pr[w_{-\tilde{\jmath}}=1]-\Pr[w=1]=2^{-\nu z_{-\tilde{\jmath}}}-2^{-\nu z}\le 2^{-\nu(z-1)}-2^{-\nu z}\le 1-2^{-\nu}\le \nu.$$

For each round of protocol Π, $T^{(3)}$ and T repeatedly perform the same task (completing partial transcript of S) until $w=1$ or $w_{-\tilde{\jmath}}=1$, respectively. Therefore the statistical difference between the two processes is at most $k\nu$. ☐

Combining Lemmas 13 and 14, we see that T fails to break the soundness of Π with probability at most

$$\frac{3}{m}\left(\frac{-\log \nu^2}{\nu}+4\right)+\frac{3}{10}+O\left(\frac{1}{\sqrt{\log n}}\right)+k\nu.$$

By setting $\nu=1/\sqrt{km}$, the expression becomes

$$3\sqrt{\frac{k}{m}}\log(km)+\frac{12}{m}+\frac{3}{10}+O\left(\frac{1}{\sqrt{\log n}}\right)+\sqrt{\frac{k}{m}}.$$

Since $m\ge k^2\log^2 k\log^2 n$, we conclude that T fails with probability at most $3/10+o(1)$. That is, T succeeds with nonnegligible probability, contradicting the soundness of Π. ☐

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

5.1. Commitment schemes. Commitment protocols allow a *sender* to commit itself to a value while keeping it secret from the *receiver*; this property is called *hiding*. At a later time, the commitment can be opened only to a single value as determined during the commitment protocol; this property is called *binding*. Commitment schemes come in two different flavors, statistically binding and statistically hiding; we make use of statistically binding commitments only in this paper. Below we sketch the properties of a statistically binding commitment; full definitions can be found in [20].

In *statistically binding commitments*, the binding property holds against unbounded adversaries, while the hiding property holds only against computationally bounded (nonuniform) adversaries. The statistical-binding property asserts that, with overwhelming probability over the randomness of the receiver, the transcript of the interaction fully determines the value committed to by the sender. The computational-hiding property guarantees that the commitments to any two different values are computationally indistinguishable.

PROTOCOL BOUNDEDCONCZK

Common input: An instance x of a language $L \in NP$ and a parameter k.

Stage one: For i from 1 to $2k^3$:

$P \to V$: Commit to a random bit p_i using a statistically binding commitment.

$V \to P$: Reply with a random bit v_i.

Stage two: A 4-round public-coin witness indistinguishable proof (e.g., parallel repetitions of the Blum Hamiltonicity protocol [8]) of the NP statement:

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

FIG. 2. *Our public-coin black-box bounded concurrent ZK protocol.*

Noninteractive statistically binding commitment schemes can be constructed using any one-to-one one-way function (see section 4.4.1 of [20]). Allowing some minimal interaction (in which the receiver first sends a single random initialization message), statistically binding commitment schemes can be obtained from any one-way function [33, 24].

5.2. A bounded concurrent public-coin ZK protocol. Our construction of BOUNDEDCONCZK is similar in spirit to the concurrent ZK protocol of [37]. Given a language $L \in NP$ 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 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* (note that unlike [37], the verifier does not commit to the bits v_i). 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 instantiated with a parallel repetition of the Blum Hamiltonicity protocol [8] with 2-round statistically binding commitments constructed from one-way functions. The verifier accepts if the prover is successful with the stage two proof. See Figure 2 for a complete description of protocol BoundedConcZK.

We choose $2k^3$ rounds of interaction in stage one of BOUNDEDCONCZK 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 ZK 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 ZK. We remark that k^3 was chosen for the sake of simplicity and is not optimized. We show completeness and soundness below.

BOUNDEDCONCZK is clearly complete. A prover given a correct problem instance and witness pair, $(x \in L, w)$, can commit to random bits in stage one, and use w to successfully complete the stage two proof.

We next show that BOUNDEDCONCZK has negligible soundness error. Suppose $x \notin L$. Then there are two ways for the prover to mislead the verifier:

1. The prover may have $p_i = v_i$ for $k^3 + k^2/2$ (or more) values of i either by breaking the binding property of the commitment, or by guessing luckily. The former occurs with negligible probability since the commitment is statistically binding. The latter occurs with probability $e^{-k/4}$ by the Chernoff bound.[5]
2. Otherwise, the prover may break the soundness of the stage two proof, which occurs with probability at most 2^{-k} due to the parallel repetitions.

Since $k = \omega(\log n)$, both $e^{-k/4}$ and 2^{-k} are negligible in n.

5.3. Black-box bounded concurrent ZK. We construct a black-box simulator S such that given an adversarial 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^*. Toward that goal, S employs a simple greedy strategy to incrementally generate and *fix* 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 [32], but since our protocol is public-coin, the actual analysis is quite different. Additionally, we take care to always simulate the stage two proof in a straight-line fashion without rewinds, so that we may use a simple hybrid argument to show the ZK property.

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

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 generate the prover messages in the stage two proof.

When receiving a verifier message,

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 partial view. If yes, S simply continues, "giving up" on this guess. Otherwise, S checks if $p_i^j = v_i^j$. If yes, S extends the fixed partial view up to message v_i^j and increments c^j; in this case we say that v_i^j is *rigged*. If $p_i^j \neq v_i^j$, then S rewinds V^* to start a fresh continuation from the previously fixed partial view;
4. if S receives the second stage two verifier message from any session (e.g., the challenge message of the Blum Hamiltonicity protocol), then it extends the fixed partial view up to the just received verifier message. As a consequence, all stage two proofs are simulated by S in a straight-line fashion without rewinds;
5. if S has performed $k - 1$ rewinds without rigging a message or encountering a stage two verifier message, and on the kth try again receives $v_i^j \neq p_i^j$, where p_i^j is not fixed and $c^j < 2k^2$, then S simply gives up and pretends to *rig* v_i^j anyway (albeit incorrectly). That is, S extends the fixed partial view up to message v_i^j and increments c^j.

[5]Here we use the following form of Chernoff bound. If $\{X_i\}$ are i.i.d. satisfying $\Pr[X_i = 0] = \Pr[X_i = 1] = 1/2$, then $\Pr[\sum_{i=1}^n X_i \geq n/2 + a] \leq e^{-2a^2/n}$

The next two lemmas show that S is a k-bounded black-box ZK simulator when $k \in \omega(\log n)$.

LEMMA 15. *S runs in (strict) polynomial time.*

Proof. S performs at most $km(2k^2)$ rewinds, which is polynomial in n. ∎

LEMMA 16. *If $x \in L$ and $m \leq k$, then $S^{V^*}(x, z)$ and $\mathrm{View}^P_{V^*}(x, z)$ are computationally indistinguishable over n.*

Proof. We introduce a series of hybrids.

Hybrid 1. Our first hybrid S_1 is given witness w to the statement $x \in L$. S_1 proceeds identically as S until a stage two proof is reached. S_1 aborts if S aborts, but uses the witness w instead of the various p_i's to complete the stage two proof. Even though S performs many rewinds, S never rewinds a partial stage two proof. Therefore, $S^{V^*}(x, z)$ and $S_1^{V^*}(x, z)$ are computationally indistinguishable because the stage two proof is witness indistinguishable.

Hybrid 2. Our second hybrid S_2 is identical to S_1 except that it samples two random bits for each stage one commitment p_i and q_i. S_2 commits to p_i, but checks v_i against q_i. Since S_1 gives polynomially many commitments and runs in polynomial time, and since each commitment is computationally hiding and independent from the rest of the execution of S_1 (stage two proofs are provided using w), $S_1^{V^*}(x, z)$ and $S_2^{V^*}(x, z)$ are computationally indistinguishable.

Hybrid 3. Our third hybrid S_3 is identical to S_2 except that S_3 *always* gives a stage two proof using witness w even if S_2 aborts. To see that $S_2^{V^*}(x, z)$ and $S_3^{V^*}(x, z)$ are computationally indistinguishable, it suffices to show that S_2 aborts with negligible probability.

Observe that whenever S extends the fixed partial view (either by rigging a commitment, or by encountering a verifier challenge in a stage two proof), at most one commitment from each session with fewer than $2k^2$ rigged messages is fixed as part of the simulator output. This is because before encountering a second commitment in any session, S would first try to rig the first commitment. For each session, S rigs at most $2k^2$ stage one commitments and encounters at most one stage two verifier challenge. Therefore, the number of commitments fixed per session without rigging is at most $(k-1)(2k^2+1) = 2k^3 - (2k^2 - k + 1)$. In other words, every session will have at least $2k^2 - k + 1$ commitments rigged.

We now show that except with negligible probability, S_2 will have $k^3 + k^2/2$ correct guesses per session. Recall that the guesses of S_2, q_i, are independent from V^*'s responses since these guesses play no part in the commitments sent to V^*. Therefore, except with probability $\mathrm{poly}(n)2^{-k}$, every rigged commitment is a correct guess. Next, for the $2k^3 - (2k^2 - k + 1) \geq 2k^3 - 2k^2$ messages that are not rigged, we apply the Chernoff bound to see that except with probability $e^{-O(k)}$, we should have at least $(k^3 - k^2) - k^2/4 = k^3 - 5k^2/4$ correct guesses. Thus, except with negligible probability,[6] we have a total of $(k^3 - 5k^2/4) + (2k^2 - k + 1) \geq k^3 + k^2/2$ correct guesses, as desired.

Final step. S_3 is now identical to P (sends identically distributed messages) except that it may rewind V during the execution. But S_3 rewinds only if $q_i \neq v_i$, an event independent from the protocol execution. Therefore $S_3^{V^*}(x, z)$ is *identical* to $\mathrm{View}^P_{V^*}(x, z)$. This concludes the proof. ∎

[6]Recall again that 2^{-k} and $e^{-O(k)}$ are negligible in n since $k = \omega(\log n)$.

6. Application to resettably sound arguments. In this section we show how to achieve more general notions of resettable soundness that were not required for our main theorem. First, we need an argument of knowledge as a building block.

6.1. Proofs and arguments of knowledge. Loosely speaking, an interactive proof is a proof of knowledge if the prover convinces the verifier that it *possesses*, or can *feasibly compute*, a witness for the statement proved.

DEFINITION 6 (proof of knowledge [6]). *An interactive protocol* $\Pi = \langle P, V \rangle$ *is a* proof of knowledge *(resp.,* argument of knowledge*) of language* L *with respect to witness relation* R_L *if* Π *is indeed an interactive proof (resp., argument) for* L*. Additionally, there exists a polynomial* q*, a negligible function* ν*, and a probabilistic oracle machine* E*, such that for every interactive machine* P^* *(resp., polynomially sized machine* P^**) and every* $x \in L$*, the following holds:*

1. *If* $\Pr[\langle P^*, V \rangle (x) = 1] > \nu(|x|)$*, then on input* x *and oracle access to* $P^*(x)$*, machine* E *outputs a string from the* $R_L(x)$ *within an expected number of steps bounded by*

$$\frac{q(|x|)}{\Pr[\langle P^*, V \rangle (x) = 1] - \nu(|x|)}.$$

The machine E *is called the knowledge extractor.*

6.2. Resettably sound arguments. It is shown implicitly in [16] that any constant-round public-coin argument is fixed-input resettably sound if the verifier uses a pseudorandom function to generate its messages. In [2, Proposition 3.5], the analysis is extended to show that any constant-round public-coin argument of knowledge for $L \in NP$ is a (full-blown) resettably sound argument of knowledge of L, again if the verifier uses a pseudorandom function to generate its messages. We give a pair of analogous theorems below, based on our techniques in section 3.

THEOREM 17. *Let* $\Pi = \langle P, V \rangle$ *be a public-coin argument for an NP language* L *with negligible soundness error. Define* $\tilde{\Pi}^m = \langle P^m, \tilde{V}^m \rangle$ *to be* m *parallel repetitions of* Π *with the following modification:* $\tilde{V}^m$ *will sample a pseudorandom function* f *at the beginning of the protocol and construct each verifier message by applying* f *to the prover messages received so far. Then, whenever* $m \geq k^2 \log^2 n$*,* $\tilde{\Pi}^m$ *is a fixed-input resettably sound argument.*

THEOREM 18. *Let* $\Pi = \langle P, V \rangle$ *be a public-coin argument of knowledge for an* NP *language* L *with negligible soundness error. Define* $\tilde{\Pi}^m = \langle P^m, \tilde{V}^m \rangle$ *similarly to* Theorem 17*. Then, whenever* $m \geq k^2 \log^2 n$*,* $\tilde{\Pi}^m$ *is a resettably sound argument of knowledge.*

Note that in contrast with section 3, we have replaced multiwise independent hash functions with pseudorandom functions. This is because a resettably sound argument needs to guard against all polynomial time resetting attacks, and so we cannot assume a universal bound on the running time of the attacks.

Proof sketch of Theorem 17. Suppose some polynomial time P_m^* breaks the fixed-input resettable soundness property against $\tilde{V}^m$. Let $\hat{V}^m$ be a hybrid verifier that is identical to $\tilde{V}^m$ except that $\hat{V}^m$ uses a truly random function F instead of a pseudorandom function f. Then, by the property of a pseudorandom function, P_m^* also breaks the fixed-input resettable soundness property against $\hat{V}^m$. Now, the techniques of section 3.3 show how to to construct a cheating P^* based on P_m^* that contradicts the soundness property of Π. This gives a contradiction. ∎

Proof sketch of Theorem 18. We use the same techniques as [2]. Consider using the same proof sketch as Theorem 17. It is easy to extend the techniques of section 3.3

to full-blown resettable attacks where P_m^* selects the input instances adaptively. The main subtlety, as pointed out by [2], is the hybrid argument involving the pseudorandom functions.

We need to show that if P_m^* breaks the resettable soundness property against the pseudorandom $\tilde{V}^m$, then it should also break the resettable soundness property against the truly random $\hat{V}^m$. The subtlety here is that a computationally bounded distinguisher cannot determine whether P_m^* has completed a successful resetting attack or not, because it cannot determine whether the x's chosen by P_m^* are in L or not. To overcome this obstacle, we require Π to be an argument of knowledge, i.e., there is a witness-extraction algorithm. We may then apply the witness-extraction algorithm to P^* (constructed from P_m^*) to determine whether the input instance accepted by V is indeed in the language L or not. $\square$

Acknowledgments. We thank Johan Håstad and the reviewers for their invaluable comments, and for highlighting our work's connection with resettable soundness. We are especially grateful to Oded Goldreich for his many suggestions that greatly improved the presentation.

REFERENCES

[1] L. Babai and S. Moran, *Arthur-Merlin games: A randomized proof system, and a hierarchy of complexity class*, J. Comput. System Sci., 36 (1988), pp. 254–276.
[2] B. Barak, O. Goldreich, S. Goldwasser, and Y. Lindell, *Resettably-sound zero-knowledge and its applications*, in Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, 2001, pp. 116–125.
[3] B. Barak and O. Goldreich, *Universal arguments and their applications*, in Comput. Complexity, (2002), pp. 162–171.
[4] B. Barak and Y. Lindell, *Strict polynomial-time in simulation and extraction*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 484–493.
[5] B. Barak, *How to go beyond the black-box simulation barrier*, in Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, 2001, pp. 106–115.
[6] M. Bellare and O. Goldreich, *On defining proofs of knowledge*, in Proceedings of the 12th Annual International Cryptology Conference, 1992, pp. 390–420.
[7] M. Bellare, R. Impagliazzo, and M. Naor, *Does parallel repetition lower the error in computationally sound protocols?*, in Proceedings of the 38th Annual Symposium on Foundations of Computer Science, 1997, pp. 374–383.
[8] M. Blum, *How to prove a theorem so no one else can claim it*, Proc. of the International Congress of Mathematicians, (1986), pp. 1444–1451.
[9] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali, *Resettable zero-knowledge (extended abstract)*, in Proceedings of the 32nd Annual ACM Symposium on Theory of Computing, 2000, pp. 235–244.
[10] R. Canetti, J. Kilian, E. Petrank, and A. Rosen, *Black-box concurrent zero-knowledge requires* $\tilde{\omega}(\log n)$ *rounds*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 570–579.
[11] B. Chor and O. Goldreich, *On the power of two-point based sampling*, J. Complexity, 5 (1989), pp. 96–106.
[12] I. Damgård, *Efficient concurrent zero-knowledge in the auxiliary string model*, in Proceedings of the 2000 International Conference on the Theory and Application of Cryptographic Techniques, 2000, pp. 418–430.
[13] C. Dwork, M. Naor, and A. Sahai, *Concurrent zero-knowledge*, J. ACM, 51 (2004), pp. 851–898.
[14] U. Feige, D. Lapidot, and A. Shamir, *Multiple non-interactive zero knowledge proofs based on a single random string*, in Proceedings of the 31st Annual Symposium on Foundations of Computer Science, 1990, pp. 308–317.
[15] U. Feige and A. Shamir, *Witness indistinguishable and witness hiding protocols*, in Proceedings of the 22nd Annual ACM Symposium on Theory of Computing, 1990, pp. 416–426.
[16] O. Goldreich and A. Kahan, *How to construct constant-round zero-knowledge proof systems for NP*, J. Cryptology, 9 (1996), pp. 167–190.

[17] O. Goldreich and H. Krawczyk, *On the composition of zero-knowledge proof systems*, SIAM J. Comput., 25 (1996), pp. 169–192.
[18] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems*, J. ACM, 38 (1991), pp. 691–729.
[19] O. Goldreich and Y. Oren, *Definitions and properties of zero-knowledge proof systems*, J. Cryptology, 7 (1994), pp. 1–32.
[20] O. Goldreich, *Foundations of Cryptography—Basic Tools*, Cambridge University Press, Cambridge, UK, 2001.
[21] O. Goldreich, *Concurrent zero-knowledge with timing, revisited*, in Proceedings of the 34th Annual ACM Symposium on Theory of Computing, 2002, pp. 332–340.
[22] S. Goldwasser, S. Micali, and C. Rackoff, *The knowledge complexity of interactive proof systems*, SIAM J. Comput., 18 (1989), pp. 186–208.
[23] I. Haitner, A. Rosen, and R. Shaltiel, *On the (im)possibility of Arthur-Merlin witness hiding protocols*, in Proceedings of the 6th Theory of Computing Conference, 2009, pp. 220–237.
[24] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, *A pseudorandom generator from any one-way function*, SIAM J. Comput., 28 (1999), pp. 1364–1396.
[25] J. Håstad, R. Pass, D. Wikström, and K. Pietrzak, *An efficient parallel repetition theorem*, in Proceedings of the 7th Theory of Computing Conference, 2010, pp. 1–18.
[26] T. Holenstein, *Parallel repetition: Simplifications and the no-signaling case*, in Proceedings of the 39th Annual ACM Symposium on Theory of Computing, 2007, pp. 411–419.
[27] R. Impagliazzo, R. Jaiswal, and V. Kabanets, *Chernoff-type direct product theorems*, in Proceedings of the 27th Annual International Cryptology Conference, 2007, pp. 500–516.
[28] R. Impagliazzo and A. Wigderson, *$P = BPP$ if E requires exponential circuits: Derandomizing the xor lemma*, in Proceedings of the 29th Annual ACM Symposium on Theory of Computing, 1997, pp. 220–229.
[29] J. Katz, *Which languages have* 4*-round zero-knowledge proofs?*, in Proceedings of the 5th Theory of Cryptography, 2008, pp. 73–88.
[30] J. Kilian, E. Petrank, and C. Rackoff, *Lower bounds for zero knowledge on the internet*, in Proceedings of the 39th Annual Symposium on Foundations of Computer Science, 1998, pp. 484–492.
[31] J. Kilian and E. Petrank, *Concurrent and resettable zero-knowledge in poly-loalgorithm rounds*, in Proceedings of the 33rd Annual ACM Symposium on Theory of Computing, 2001, pp. 560–569.
[32] Y. Lindell, *Bounded-concurrent secure two-party computation without setup assumptions*, in Proceedings of the 35th Annual ACM Symposium on Theory of Computing, 2003, pp. 683–692.
[33] M. Naor, *Bit commitment using pseudorandomness*, J. Cryptology, 4 (1991), pp. 151–158.
[34] R. Pass and M. Venkitasubramaniam, *An efficient parallel repetition theorem for Arthur-Merlin games*, in Proceedings of the 39th Annual ACM Symposium on Theory of Computing, 2007, pp. 420–429.
[35] M. Prabhakaran, A. Rosen, and A. Sahai, *Concurrent zero knowledge with logarithmic round-complexity*, in Proceedings of the 43rd Annual Symposium on Foundations of Computer Science, 2002, pp. 366–375.
[36] R. Raz, *A parallel repetition theorem*, SIAM J. Comput., 27 (1998), pp. 763–803.
[37] R. Richardson and J. Kilian, *On the concurrent composition of zero-knowledge proofs*, in Proceedings of the 1999 International Conference of the Theory and Application of Cryptographic Techniques, 1999, pp. 415–432.
[38] A. Rosen, *A note on the round-complexity of concurrent zero-knowledge*, in Proceedings of the 20th Annual International Cryptology Conference, 2000, pp. 451–468.