SIAM J. COMPUT.
Vol. 25, No. 1, pp. 169–192, February 1996

ON THE COMPOSITION OF ZERO-KNOWLEDGE PROOF SYSTEMS*

ODED GOLDREICH† AND HUGO KRAWCZYK‡

Abstract. The wide applicability of zero-knowledge interactive proofs comes from the possibility of using these proofs as subroutines in cryptographic protocols. A basic question concerning this use is whether the (sequential and/or parallel) composition of zero-knowledge protocols is zero-knowledge too. We demonstrate the limitations of the composition of zero-knowledge protocols by proving that the original definition of zero-knowledge is not closed under sequential composition; and that even the strong formulations of zero-knowledge (e.g., black-box simulation) are not closed under parallel execution.

We present lower bounds on the round complexity of zero-knowledge proofs, with significant implications for the parallelization of zero-knowledge protocols. We prove that three-round interactive proofs and constant-round Arthur–Merlin proofs that are black-box simulation zero-knowledge exist only for languages in BPP. In particular, it follows that the "parallel versions" of the first interactive proofs systems presented for quadratic residuosity, graph isomorphism, and any language in NP, are not black-box simulation zero-knowledge, unless the corresponding languages are in BPP. Whether these parallel versions constitute zero-knowledge proofs was an intriguing open questions arising from the early works on zero-knowledge. Other consequences are a proof of optimality for the round complexity of various known zero-knowledge protocols and the necessity of using secret coins in the design of "parallelizable" constant-round zero-knowledge proofs.

Key words. zero-knowledge, cryptographic protocols, interactive proofs

AMS subject classifications. 68Q99, 94A60

1. Introduction.

In this paper we investigate the problem of composing zero-knowledge proof systems. Zero-knowledge proof systems, introduced in the seminal paper of Goldwasser, Micali, and Rackoff [GMR1], are efficient interactive proofs which have the remarkable property of yielding nothing but the validity of the assertion. Namely, whatever can be efficiently computed after interacting with a zero-knowledge prover, can be efficiently computed on input of a valid assertion. Thus, a zero-knowledge proof is computationally equivalent to an answer of a trusted oracle.

A basic question regarding zero-knowledge interactive proofs is whether their composition remains zero-knowledge. This question is not only of theoretical importance, but is also crucial to the utilization of zero-knowledge proof systems as subprotocols inside cryptographic protocols. Of particular interest are sequential and parallel composition. Candidate "theorems" (whose correctness we investigate) are listed here.

Sequential Composition. Let Π_1 and Π_2 be zero-knowledge proof systems for languages L_1 and L_2, respectively. Then, on input $x_1 \circ x_2$, first executing Π_1 on x_1 and afterwards executing Π_2 on x_2 constitutes a zero-knowledge interactive proof system for $L_1 \circ L_2$.

Parallel Composition. Let Π_1 and Π_2 be as above. Then, on input $x_1 \circ x_2$, concurrently executing Π_1 on input x_1 and Π_2 on x_2 constitutes a zero-knowledge interactive proof system for $L_1 \circ L_2$. (Concurrent execution means that the ith message of the composed protocol consists of the concatenation of the ith messages in Π_1 and Π_2, respectively.)

Sequential composition. Soon after the publication of [GMR1], several researchers noticed that the formulation of zero-knowledge proposed therein (hereafter referred as the *original* formulation) is probably not closed under sequential composition. In particular, Feige

*Received by the editors August 30, 1993; accepted for publication (in revised form) July 28, 1994. This research was partially supported by the Fund for Basic Research Administered by the Israeli Academy of Sciences and Humanities. A preliminary version of this paper appeared in the *Proc. 17th ICALP*, Lecture Notes in Computer Science, Vol. 443, Springer-Verlag, Berlin, 1990, pp. 268–282.

†Department of Computer Science, Technion, Haifa, Israel.

‡IBM T. J. Watson Research Center, Yorktown Heights, NY 10598 (hugo@watson.ibm.com). This research work was performed while this author was with the Department of Computer Science, Technion, Haifa, Israel.

and Shamir [Fei] proposed a protocol conjectured to be a counterexample to the Sequential Composition "Theorem." In this paper we use the ideas of [Fei] and new results on pseudorandom distributions [GK] to prove that, indeed, the *original formulation of zero-knowledge is not closed under sequential composition.* Our proof is independent of any intractability assumption. It applies to the notion of *computational* zero-knowledge (see §2), and uses computationally unbounded provers. (So far no proof exists for the same result with provers limited to polynomial time, or for statistical or perfect zero-knowledge.)

The reader should be aware that the Sequential Composition Theorem was proven (by Goldreich and Oren [GO], [Ore]) for a stronger ("nonuniform") formulation of zero-knowledge suggested by several authors (cf. [Fei], [GMR2], [GO], [Ore], and [TW]). The Sequential Composition Theorem is crucial to the utilization of zero-knowledge interactive proofs in cryptographic applications and in particular to the construction of cryptographic protocols for playing any computable game [Yao], [GMW2].

Parallel composition. Parallel composition of interactive proofs is widely used as a means of decreasing the error probability of proof systems, while maintaining the number of rounds. Of course, one would be interested in applying these advantages of parallelism to zero-knowledge protocols as well. Parallelism is also used in multiparty protocols in which parties wish to prove (the same and/or different) statements to various parties concurrently. Unfortunately, we show in this paper a counterexample to the Parallel Composition Theorem. Namely, we introduce a pair of protocols which are (computational) zero-knowledge with respect to the strongest known definitions (including the nonuniform formulation discussed above and the "black-box simulation" formulation discussed below) yet their parallel composition is not zero-knowledge (not even in the "weak" sense of the original [GMR1] formulation). Also in this case, our proof does not rely on any unproven hypotheses; on the other hand, it uses in an essential way the unbounded computational power of the prover and the computational notion of zero-knowledge. Based on intractability assumptions, Feige and Shamir [FS2] show a perfect zero-knowledge protocol with a polynomial-time prover which fails parallel composition. Our results below on three-round zero-knowledge proofs imply a similar result, but our case requires a superlogarithmic number of repetitions, while in [FS2] two repetitions suffice.

By the above result we have ruled out the possibility of proving that particular interactive proofs are zero-knowledge by merely arguing that they are the result of parallel composition of various zero-knowledge protocols. But this does not resolve the question of whether concrete cases of composed interactive proofs are zero-knowledge. In particular, since the first presentation of the results in [GMR1] and [GMW1], it was repeatedly asked whether the "parallel versions" of the interactive proofs presented for quadratic residuosity, graph isomorphism, and any language in NP are zero-knowledge.

Round complexity of zero-knowledge proofs. In this paper we prove a general result concerning the round complexity of zero-knowledge interactive proofs which, in particular, resolves the question of parallelization of the above-mentioned protocols. This general result states that *only BPP languages have three-round interactive proofs which are black-box simulation zero-knowledge.*[1] Since the parallel versions of the above examples are three-round interactive proofs (with negligible cheating probability for the prover) it follows that these interactive proofs cannot be proven zero-knowledge using black-box simulation zero-knowledge, unless the corresponding languages are in BPP. This (negative) result is proven for computa-

[1]This result applies to interactive proofs in which the prover can convince the verifier of accepting a false assertion with only negligible probability. The above-mentioned languages have three-round zero-knowledge interactive proofs in which the prover has a significant (e.g., constant) probability of cheating.

tional zero-knowledge proofs and therefore applies to statistical and perfect zero-knowledge as well.

Loosely speaking, we say that an interactive proof for a language L *is black-box simulation zero-knowledge* if there exists a (probabilistic polynomial-time) universal simulator which, using any (even nonuniform) verifier V^* as a black box, produces a probability distribution which is polynomially indistinguishable from the distribution of conversations of (the same) V^* with the prover, on inputs in L. This definition of zero-knowledge is more restrictive than the original one, which allows each verifier V^* to have a specific simulator S_{V^*}. Nevertheless, all *known* zero-knowledge protocols are also black-box simulation zero-knowledge. This fact cannot come as a surprise since it is hard to conceive of a way of taking advantage of the full power of the more liberal definition.

It is not plausible that our result could be extended to four-round interactive proofs since such proofs are known for languages believed to be outside BPP. The protocols for quadratic nonresiduosity [GMR1] and graph nonisomorphism [GMW1] are such examples. In addition, zero-knowledge interactive proofs of five rounds are known for quadratic residuosity and graph isomorphism [BMO1], and, assuming the existence of claw-free permutations, there exist five-round zero-knowledge interactive proofs for any language in NP [GKa]. Moreover, our results extend to zero-knowledge *arguments*[2], for which Feige and Shamir [FS1] presented (assuming the existence of one-way functions) a four-round protocol for any language in NP. Our result implies that the round complexity of this protocol is optimal (as long as BPP $\neq$ NP).

Constant-round Arthur–Merlin proofs. When restricting ourselves to Arthur–Merlin interactive proofs, we can extend the above result to any constant number of rounds. We show that *only BPP languages have constant-round Arthur–Merlin proofs which are also black-box simulation zero-knowledge.*

Arthur–Merlin interactive proofs, introduced by Babai [Bab], are interactive proofs in which all the messages sent by the verifier are the outcome of his coin tosses. In other words, the verifier "keeps no secrets from the prover." Our result is a good reason to believe that the only feasible way of constructing constant-round zero-knowledge interactive proofs is to let the verifier use "secret coins." Indeed, the above-mentioned constant-round zero-knowledge proofs, as well as the constant-round statistical zero-knowledge proofs of [BMO2], use secret coins. Thus, secret coins do help in the zero-knowledge setting. This should be contrasted with the result of Goldwasser and Sipser [GS], which states that Arthur–Merlin interactive proofs are *equivalent* to general interactive proofs (as far as language recognition is concerned). They show that any language having a general interactive proof (IP) of k rounds also has an Arthur–Merlin (or AM) proof of k rounds. Using our result we see that in the zero-knowledge setting such a preservation of rounds (when transforming IP into AM) is not plausible (e.g., graph nonisomophism).

Our result concerning Arthur–Merlin proofs is tight in the sense that the languages considered above (e.g., graph nonisomorphism, every language in NP) have unbounded (i.e., $\omega(n)$-round, for every unbounded function $\omega : N \rightarrow N$) Arthur–Merlin proof systems which are black-box simulation zero-knowledge. In particular, we get that bounded-round Arthur–Merlin proofs which are black-box zero-knowledge exist only for BPP, while unbounded round proofs of the same type exist for all PSPACE (if one-way functions exist [IY], [B*], [Sha]). That is, while the *finite hierarchy* of languages having black-box zero-knowledge Arthur–

[2]Interactive *arguments* (also known as "computationally sound proofs" and "computationally convincing protocols") differ from an interactive proof system in that the soundness condition of the system is formulated with respect to *probabilistic polynomial-time* provers, possibly with auxiliary input (see [BCC]). Namely, *efficient* provers cannot fool the verifier into accepting an input not in the language, except with negligible probability.

Merlin proofs collapses to BPP (= AM(0)), the corresponding *infinite hierarchy* contains all of PSPACE. This implies (assuming the existence of one-way functions) a separation between the two hierarchies.

Organization. In §2 we outline the definitions of interactive proofs and zero-knowledge, and introduce some terminology and notation used throughout the paper. Section 3 presents the definitions and results concerning pseudorandom distributions that are used for disproving the composition theorems. In §§4 and 5 we present these disproofs for the case of sequential and parallel composition, respectively. Finally, in §6 we present the lower bounds on the round complexity of black-box simulation zero-knowledge proofs. We stress that this last section is technically independent from §§3, 4, and 5 and can be read without going through these sections.

2. Preliminaries.

The notions of interactive proofs and zero-knowledge were introduced by Goldwasser, Micali, and Rackoff [GMR1]. Here, we give an informal outline of these notions. For formal and complete definitions, as well as the basic results concerning these concepts, the reader is referred to [GMR1], [GMW1], and [GMR2].

An *interactive proof* is a two-party protocol in which a computationally unrestricted *prover*, P, interacts with a probabilistic polynomial-time *verifier*, V, by exchanging messages. Both parties share a common input x. At the end of the interaction the verifier computes a predicate depending on this input and the exchanged messages in order to *accept* or *reject* the input x. Such a protocol, denoted $\langle P, V\rangle$, is called an *interactive proof for a language L* if the following two conditions hold.

Completeness property. For any positive constant c and sufficiently long $x \in L$, Prob(V accepts x) $> 1 - |x|^{-c}$.

Soundness property. For any positive constant c and sufficiently long $x \notin L$, Prob(V accepts x) $< |x|^{-c}$, no matter how the prover behaves during the protocol.

(The above probabilities are taken over the coin tosses of both the prover and the verifier.) In other words, we require that on inputs belonging to L the probability that the prover P "convinces" V to accept the common input is almost 1, while on inputs outside L there is no prover that can fool V into accepting, except with negligible probability.

Note. Notice that we define an interactive proof to have a negligible probability of error. Some authors define this probability to be just a constant (e.g., $\frac{1}{3}$). The latter is motivated by the fact that constant-error interactive proofs can be converted into negligible-error proofs by parallel repetition. However, in the setting of zero-knowledge interactive proofs, our results show that such parallel repetition may sacrifice the zero-knowledge property.

An interactive proof in which the *honest* verifier chooses all its messages at random (i.e., with uniform probability over the set of all strings of the same length as the message) is called an *Arthur–Merlin* interactive proof [Bab]. That is, in an Arthur–Merlin proof system the only nontrivial computation carried out by the honest verifier is the evaluation of a polynomial-time predicate at the end of the interaction, in order to decide the acceptance or rejection of the input to the protocol. We say that such a verifier uses *public coins*. (Notice that there is no "public coin" restriction on the cheating verifiers.)

We say that an interactive proof has k rounds if there are a total of k messages (alternately) exchanged between the prover and verifier during the protocol (i.e., we count messages from both parties). In general, the number k can be a function $k(|x|)$ of the input length. The notation IP(k) stands for the class of languages having k-round interactive proofs, and AM(k) stands for languages having k-round Arthur–Merlin interactive proofs.

An interactive proof is called *zero-knowledge* if on input $x \in L$ no probabilistic polynomial-time verifier (i.e., one that may arbitrarily deviate from the predetermined program) gains information from the execution of the protocol except the knowledge that x belongs to L.

This means that any polynomial-time computation based on the conversation with the prover can be simulated, without interacting with the real prover, by a probabilistic polynomial-time machine ("the simulator") that gets x as its only input. More precisely, let $\langle P, V^* \rangle(x)$ denote the probability distribution generated by the interactive machine (verifier) V^*, which interacts with the prover P on input $x \in L$. We say that an interactive proof is *zero-knowledge* if for all probabilistic polynomial-time machines V^*, there exists a probabilistic expected polynomial-time algorithm M_{V^*} (called the *simulator*) that on inputs $x \in L$ produces probability distributions $M_{V^*}(x)$ that are polynomially indistinguishable from the distributions $\langle P, V^* \rangle(x)$. (This notion of zero-knowledge is also called *computational zero-knowledge*.)[3]

The above formalization of the notion of zero-knowledge is taken from the original paper by Goldwasser, Micali, and Rackoff [GMR1]. Later, stronger formulations of zero-knowledge were introduced in which the simulation requirement is extended to deal with stronger verifiers [Fei], [GMR2], [GO], [Ore], [TW], namely, verifiers with nonuniform properties (e.g., probabilistic polynomial-time verifiers that get an additional *auxiliary-input* tape), or verifiers modeled by polynomial-size circuits.

One further formulation of zero-knowledge is called *black-box simulation zero-knowledge* [GO], [Ore]. This formulation differs from the former by requiring the existence of a ("universal") simulator that, using any (nonuniform) verifier V^* as a black box, succeeds in simulating the $\langle P, V^* \rangle$ interaction. In other words, there exists a probabilistic expected polynomial-time oracle machine M such that for any polynomial-size verifier V^* and for $x \in L$, the distributions $\langle P, V^* \rangle(x)$ and $M^{V^*}(x)$ are polynomially indistinguishable.

Remark. A complete formalization of the notion of black-box simulation zero-knowledge requires dealing with the following technical problem. The simulator uses V^* as a black box. This means that the simulator is responsible, during the simulation process, for feeding into the black box the external parameters that determine the behavior of V^*. These parameters are the string representing the input to the protocol, the contents of the random tape of V^*, and the messages of the prover. A problem arises when feeding the random coins used by V^*. Although the number of coin tosses used by a particular verifier V^* is bounded by a polynomial, there is no *single* polynomial that bounds this number for *all* possible verifiers. On the other hand, the simulator M runs (expected) time that is bounded by a specific polynomial. So, how can this simulator manage to feed a verifier requiring more coin tosses than this bound? In [BMO2] this problem is overcome by stating the existence of two random tapes for M. The first is used in the regular way for M's computations. The second can be entirely fed by M into V^* in a single step. That is, M can feed the random coins for the black box in an "intelligent way" as long as the number of coins does not exceed the time capability of M; beyond this number it can only feed truly random bits. We stress that this formalization is general enough to include all *known* zero-knowledge proofs.

An alternative solution to the above problem is to have, *for each* polynomial p, a simulator M_p which simulates all verifiers V^* that use at most $p(|x|)$ random coins on any input x. Clearly, the running time of the simulator M_p may depend on the polynomial p, and then the above difficulty is overcome. This second formulation is weaker than the one proposed in [BMO2], but it suffices for the results proved in our paper and is therefore adopted here. (In fact, our results in §6 only require the existence of a simulator that simulates deterministic verifiers, i.e., M_p with $p \equiv 0$.)

Based on the above remark we give our definition of black-box simulation zero-knowledge.

[3]Other definitions were proposed in which it is required that the distribution generated by the simulator be *identical* to the distribution of conversations between the verifier and the prover (*perfect* zero-knowledge), or at least statistically close (*statistical* zero-knowledge). See [GMR2] for further details.

DEFINITION. *An interactive proof* $\langle P, V\rangle$ *is called* black-box simulation zero-knowledge *if for every polynomial* p*, there exists a probabilistic expected polynomial-time oracle machine* M_p *such that for any polynomial-size verifier* V^* *that uses at most* $p(n)$ *random coins on inputs of length* n*, and for* $x \in L$*, the distributions* $\langle P, V^*\rangle(x)$ *and* $M_p^{V^*}(x)$ *are polynomially indistinguishable.*

Note. The notion of polynomial indistinguishability in the above definition can be formalized based on nonuniform polynomial-size distinguishers or uniform polynomial-time distinguishers which have black-box access to the corresponding V^*. Our results apply to both formalizations.

Terminology. Throughout this paper we use the term *negligible* to denote functions that are (asymptotically) smaller than 1 over any polynomial, and the term *nonnegligible* to denote functions that are greater than 1 over some fixed polynomial. More precisely, a function f from nonnegative integers to reals is called *negligible* if for all constants c and sufficiently large n, $f(n) < n^{-c}$. The function f is called *nonnegligible* if there exists a constant c such that for all (sufficiently large) n, $f(n) > n^{-c}$. (Observe that nonnegligible is not the complement of negligible.)

Notation. We use the notation $e \in_R S$ for "the element e is chosen with uniform probability from the set S."

3. On evasive and pseudorandom sets.

In the demonstration of counterexamples for the "composition theorems" we make use of pseudorandom distributions which have some interesting "evasiveness" properties. These properties and the corresponding proofs are given in [GK] and cited here without proof.

Roughly speaking, a distribution on a set of strings of length k is *pseudorandom* if this distribution cannot be efficiently (i.e., in time polynomial in k) distinguished from the uniform distribution on the set of all strings of length k. In order to formalize this notion one has to define it in asymptotical terms and refer to collections of distributions (called *pseudorandom ensembles*), rather than single distributions. The notion of a "pseudorandom set" is made precise in the following definition.

DEFINITION 3.1. *A set* $S \subseteq \{0,1\}^k$ *is called* $(\tau(k), \varepsilon(k))$-pseudorandom *if for any (probabilistic) circuit* C *of size* $\tau(k)$ *with* k *inputs and a single output*

$$|p_C(S) - p_C(\{0,1\}^k)| \le \varepsilon(k),$$

where $p_C(S)$ *(resp.,* $p_C(\{0,1\}^k)$*) denotes the probability that* C *outputs* 1 *when given elements of* S *(resp.,* $\{0,1\}^k$*), chosen with uniform probability.*

Note that a collection of uniform distributions on a sequence of sets $S_1, S_2, \ldots$, where each S_k is a $(\tau(k), \varepsilon(k))$-pseudorandom set, constitutes a pseudorandom ensemble, provided that both functions $\tau(n)$ and $\varepsilon^{-1}(n)$ grow faster than any polynomial. Therefore, we shall refer to such a sequence of sets as a *pseudorandom ensemble.*

We now present the concept of "evasive sets." Informally, evasiveness means that it is hard, for efficient algorithms, to find strings which belong to these sets.

DEFINITION 3.2. *Let* $S_1, S_2, \ldots$ *be a sequence of (nonempty) sets such that for every* n, $S_n \subseteq \{0,1\}^{Q(n)}$*, for a fixed polynomial* Q*. Such a sequence is called a* polynomially evasive *(denoted* P-evasive*) ensemble if for any probabilistic polynomial-time algorithm* A*, any constant* c*, any sufficiently large* n*, and any* $x \in \{O, 1\}^n$*,*

$$\text{Prob}(A(x) \in S_n) < n^{-c},$$

where the probability is taken over the random coins of algorithm A*.*

The following theorem states the existence of a P-evasive ensemble which is also pseudorandom.

THEOREM 3.1 [GK]. *There exists a* P-*evasive pseudorandom ensemble* $S_1, S_2, \dots$ *with* $Q(n) = 4n$. *Furthermore, there exists a Turing machine which on input* 1^n *outputs the set* S_n.

For disproving the parallel composition theorem we shall need a stronger notion of evasiveness. Namely, one which also resists nonuniform algorithms. This definition of evasiveness involves a collection of sets for each length, rather than a single set per length as in the uniform case.

DEFINITION 3.3. *Let* $Q(\cdot)$ *be a polynomial, and for* $n = 1, 2, \dots$ *let* $\mathbf{S}^{(n)}$ *be a collection of* 2^n *sets* $\{S_1^{(n)}, \dots, S_{2^n}^{(n)}\}$, *where each* $S_i^{(n)} \subseteq \{0, 1\}^{Q(n)}$. *The sequence* $\mathbf{S}^{(1)}, \mathbf{S}^{(2)}, \dots$ *is called a* nonuniform polynomially evasive (*denoted* P/poly-evasive) *ensemble if for any* $c > 0$, *sufficiently large* n, *and any* (*probabilistic*) *circuit* C *of size* n^c (*with* n *inputs and* $Q(n)$ *outputs*)

$$\text{Prob}(C(i) \in S_i) < \frac{1}{n^c},$$

where the probability is taken over the random coins of C *and* $i \in \{1, \dots, 2^n\}$, *both with uniform probability.*

That is, a sequence $\mathbf{S}^{(1)}$, $\mathbf{S}^{(2)}, \dots$ is a P/poly-evasive ensemble if any circuit of size polynomial in n, which gets a randomly selected index of one of the sets in $\mathbf{S}^{(n)}$, cannot succeed in outputting an element in that set, except for a negligible probability.

Remark. Notice that in the definition of P-evasive ensembles the (uniform) algorithm trying to hit an element in the evasive set S_n gets as input a string x of length n, which can be seen as an auxiliary input. The crucial difference between this "uniform" definition and the definition of P/poly-evasiveness is that in the latter the auxiliary input is allowed to be of any length polynomial in the length of the target strings, while in the former the auxiliary input is properly shorter than the target strings in the set S_n.

The following theorem states the existence of a P/poly-evasive ensemble which is composed of pseudorandom sets.

THEOREM 3.2. *There exists a* P/*poly-evasive ensemble* $\mathbf{S}^{(1)}, \mathbf{S}^{(2)}, \dots$ *with* $Q(n) = 4n$, *such that for every* n, *each* $S_i^{(n)}$ *is a* $(2^{n/4}, 2^{-n/4})$-*pseudorandom set of cardinality* 2^n. *Furthermore, there exists a Turing machine which on input* 1^n *outputs the collection* $\mathbf{S}^{(n)}$.

The proof of this theorem is given in the appendix.

4. Sequential composition of zero-knowledge protocols.

A natural requirement from the notion of zero-knowledge proofs is that the information obtained by the verifier during the execution of a zero-knowledge protocol will not enable him to extract any additional knowledge from subsequent executions of the same protocol. That is, it would be desirable for the *sequential composition* of zero-knowledge protocols to yield a protocol which is itself zero-knowledge. Such a property is crucial for applications of zero-knowledge protocols in cryptography (for details and further motivation, see [GO] and [Ore]).

We prove that the original definition of (computational) zero-knowledge introduced by Goldwasser, Micali, and Rackoff in [GMR1], *is not closed* under sequential composition. Several authors have previously observed that this definition *probably* does not guarantee the robustness of zero-knowledge under sequential composition, and hence have introduced more robust formulations of zero-knowledge [Fei], [GMR2], [GO], [Ore], [TW]. But so far, no proof has been given for the claim that computational zero-knowledge (with uniform verifiers) fails under sequential composition.

Intuitively, the reason that a zero-knowledge protocol could not be closed under sequential composition is that the definition of zero-knowledge requires that the information transmitted in the execution of the protocol is "useless" for any *polynomial-time computation*; it does not rule out the possibility that a cheating verifier could take advantage of this "knowledge" in a subsequent interaction with the (*nonpolynomial time*) prover for obtaining valuable information. This intuition (presented in [Fei]) is the basis of our example of a protocol which is zero-knowledge in a single execution but reveals significant information when composed twice in a sequence. This protocol, presented in the proof of the following theorem, uses a P-evasive ensemble as defined in Definition 3.2 and whose existence is stated in Theorem 3.1.

THEOREM 4.1. *Computational zero-knowledge* ([GMR1]*formulation*) *is not closed under sequential composition.*

Proof. Let $S_1, S_2, \ldots$ be a P-evasive pseudorandom ensemble as described in Theorem 3.1. Also, let K be an (arbitrary) hard Boolean function, in the sense that the language $L_K = \{x : K(x) = 1\}$ is not in BPP (we use this function as a "knowledge" function).

We present the following interactive-proof protocol $\langle P, V \rangle$ for the language $L = \{0,1\}^*$. (Obviously, this language has a trivial zero-knowledge proof in which the verifier accepts every input without carrying out any interaction. We intentionally modify this trivial protocol in order to demonstrate a zero-knowledge protocol which fails sequential composition.)

Let x be the common input for P and V, and let n denote the length of x. The verifier V begins by sending to the prover a random string s of length $4n$. The prover P checks whether $s \in S_n$ (the nth set in the P-evasive ensemble defined above). If this is the case (i.e., $s \in S_n$), then P sends to V the value of $K(x)$. Otherwise (i.e., $s \notin S_n$), P sends to V a string s_0 randomly selected from S_n. In any case the verifier accepts the input x (as belonging to L).

We stress that the same P-evasive ensemble is used in all the executions of the protocol. Thus, the set S_n does not depend on the specific input to the protocol, but only on its length. Therefore, the string s_0, obtained by the verifier in the first execution of the protocol, enables him to deviate from the protocol during a second execution in order to obtain the value of $K(x')$, for any x' of length n (and in particular for $x' = x$). Indeed, consider a second execution of the protocol, this time on input x'. A "cheating" verifier, which sends the string $s = s_0$ instead of choosing it at random, will get the value of $K(x')$ from the prover. Observe that this cheating verifier obtains information that it could not compute by itself. There is no way to simulate in probabilistic polynomial time the interaction in which the prover sends the value of $K(x')$; otherwise the language L_K would be in BPP (indeed, such a simulator could be used as a probabilistic polynomial-time algorithm for computing the function K with negligible error probability. To see that, notice that the real prover in an interaction with the above cheater verifier on inputs (x, x') will output $k(x')$ with probability 1. Therefore, the simulator must output the correct value of $k(x')$ with probability almost 1, or otherwise, its output is polynomially distinguishable from the real conversations). Thus, the protocol is not zero-knowledge when composed twice.

On the other hand, the protocol is zero-knowledge (when executed once). To show this, we present for any verifier V^*, a polynomial-time simulator M_{V^*} that can simulate the conversations between V^* and the prover P. There is only one message sent by the prover during the protocol. It sends the value of $K(x)$ when the string s sent by the verifier belongs to the set S_n, and a randomly selected element of S_n otherwise. By the evasivity condition of the set S_n, there is only a negligible probability that the first case holds. Indeed, no probabilistic polynomial-time machine (in our case, the verifier) can find such a string $s \in S_n$, except with negligible probability (no matter what the input x to the protocol is). Thus, the simulator can succeed by always simulating the second possibility, i.e., the sending of a random element s_0 from S_n. This step is simulated by randomly choosing s_0 from $\{0,1\}^{4n}$ rather than from S_n. The indistinguishability of this choice from the original one follows from the fact that each S_n is a pseudorandom subset of $\{0,1\}^{4n}$, and that the prover chooses s_0 from S_n with uniform probability. $\square$

Remark. The argument presented in the above proof generalizes to any language L having a zero-knowledge interactive proof. Simply modify the zero-knowledge proof for L as in the proof of Theorem 4.1.

Remark. Another example of a zero-knowledge protocol which is not closed under sequential composition was independently found by D. Simon [Sim]. His construction assumes the existence of secure encryption systems.

5. Parallel composition of zero-knowledge protocols. In this section we address the question of whether zero-knowledge interactive proofs are robust under parallel composition.

Clearly, we cannot expect the original Goldwasser–Micali–Rackoff (GMR) definition to satisfy this condition: it is easy to see that a zero-knowledge protocol which is not closed under sequential composition can be transformed into another zero-knowledge protocol which fails parallel composition.

In light of the fact that *auxiliary-input* zero-knowledge is robust under sequential composition [GO], [Ore], it is an interesting open question whether this formulation of zero-knowledge is also robust under parallel composition. The main result of this section is that this is *not* the case. We prove the existence of protocols which are zero-knowledge even against nonuniform verifiers (e.g., auxiliary-input zero-knowledge), but which do not remain zero-knowledge when executed twice in parallel. As in the case of sequential composition our results concern only computational zero-knowledge.

The ideas used for the design of a protocol which fails parallel composition are similar to those used for the sequential case. There, we have used the pseudorandomness and evasiveness of some sets to construct the intended protocol. We also use this method here. The main difficulty of extending these properties to the present case is that now we need an evasive collection which resists even nonuniform verifiers. Clearly, a P-evasive ensemble will not satisfy this condition, since for any set of strings there exist nonuniform verifiers which can output elements in this set (e.g., by getting such a string as auxiliary input). Instead, we use the notion of P/poly-evasive ensembles as defined in Definition 3.3. Based on Theorem 3.2, which states the existence of such ensembles, we prove the main result of this section.

THEOREM 5.1. *Computational zero-knowledge (even with nonuniform verifiers) is not closed under parallel composition.*

Proof. We present a pair of protocols $\langle P_1, V_1\rangle$ and $\langle P_2, V_2\rangle$ which are zero-knowledge when executed independently, but whose parallel composition is provably not zero-knowledge.

We use some dummy steps in the protocols in order to achieve synchronization between them. Of course, one can modify the protocol, substituting these extra steps by significant ones. The version we give here prefers simplicity over naturality. Both protocols consist of five steps and are described below (see also Fig. 1).

The first protocol is denoted $\langle P_1, V_2\rangle$. Let x be the input to the protocol and let n denote its length. The protocol uses (for all its executions) a P/poly-evasive ensemble $S^{(1)}, S^{(2)}, \ldots$ with the properties described in Theorem 3.2. It also involves a hard Boolean function K as in the proof of Theorem 4.1. The prover P_1 begins by sending to V_1 an index $i \in_R \{1, \ldots, 2^n\}$. After two dummy steps the verifier V_1 sends to P_1 a string $s \in_R \{0, 1\}^{4n}$. The prover P_1 checks whether $s \in S_i^{(n)}$. If this is the case then it sends to V_1 the value of $K(x)$, (otherwise an empty message). This concludes the protocol.

The second protocol $\langle P_2, V_2\rangle$ uses the *same* P/poly-evasive ensemble $S^{(1)}, S^{(2)}, \ldots$ as protocol $\langle P_1, V_1\rangle$ does. The first step of the protocol is a dummy one. In the second step the verifier V_2 sends to P_2 an index $j \in_R \{1, \ldots, 2^n\}$. The prover P_2 responds with a string $r \in_R S_j^{(n)}$. After two more dummy steps the protocol stops.

We show that each of the above protocols is indeed zero-knowledge (even for nonuniform verifiers). For the first protocol, there are two steps of the prover to be simulated. In the

P_1	V_1	step	P_2	V_2
$i \in_R \{1, \cdots, 2^n\} \rightarrow$		1	dummy step	
	dummy step	2		$\leftarrow j \in_R \{1, \cdots, 2^n\}$
dummy step		3	$r \in_R S_j^{(n)} \rightarrow$	
	$\leftarrow s \in_R \{0,1\}^{4n}$	4		dummy step
if $s \in S_i^{(n)}$: $K(x) \rightarrow$		5	dummy step	

FIG. 1. *Protocols* $\langle P_1, V_1\rangle$ *and* $\langle P_2, V_2\rangle$ *with input* x.

first step P_1 sends an index $i \in_R \{1, \dots, 2^n\}$. The simulator does the same. In the second step, the prover sends the value of $K(x)$ *only* if the verifier succeeds in presenting him with a string which belongs to the set $S_i^{(n)}$. By the evasivity condition of the sequence $S^{(1)}, S^{(2)}, \dots$, this will happen with negligible probability and therefore the simulator can always simulate this step as for the case where the verifier sends a string $s \notin S_i^{(n)}$. (Observe that the circuits in the definition of P/poly-evasive ensembles only get as input the index of the set to be hit. Nevertheless, in our case the circuits also have an additional input x. Clearly, this cannot help them find an element in $S_i^{(n)}$; otherwise, circuits which have such a string incorporated will contradict the evasiveness condition.)

In the second protocol, $\langle P_2, V_2\rangle$, the only significant step of the prover P_2 is when it sends an element $r \in_R S_j^{(n)}$ in response to the index j sent by the verifier. In this case the simulator will send a string $r' \in_R \{0, 1\}^{4n}$. Using the pseudorandomness property of the set $S_j^{(n)}$ we get that the simulator's choice is polynomially indistinguishable from the prover's one.

Finally, we show that the parallel composition of the above protocols into a single protocol $\langle P, V\rangle$ is not zero-knowledge. Let V^* be a "cheating" verifier which behaves as follows. Instead of sending a randomly selected index j (corresponding to the second step of the subprotocol $\langle P_2, V_2\rangle$) it sends the index i received from P as part of P_1's first step. Thus, $j = i$, and the prover P will respond with a string $r \in S_i^{(n)}$. In the next step this V^* will send string r to P instead of the "random" string s that V_1 should send to P_1. The prover P will verify that $r \in S_i^{(n)}$ and then will send the information $K(x)$. By the hardness of the function K this step cannot be simulated by a probabilistic polynomial-time machine. Therefore, the composed protocol $\langle P, V\rangle$ is not zero-knowledge. $\square$

Remark. The two protocols $\langle P_1, V_1\rangle$ and $\langle P_2, V_2\rangle$ can be merged into a single zero-knowledge protocol which is not robust under parallel composition. In this merged protocol, the verifier chooses (at random) an index $i \in \{1, 2\}$, sends it to the prover, and then both parties execute the protocol $\langle P_i, V_i\rangle$. When executing two copies of this protocol in parallel, the verifiers may choose $i = 1$ and $i = 2$, respectively, thus forcing a parallel execution of $\langle P_1, V_1\rangle$ and $\langle P_2, V_2\rangle$, which we have shown not to be zero-knowledge.

6. On the round complexity of zero-knowledge proofs.

In this section we present lower bounds on the round complexity of black-box simulation zero-knowledge interactive proofs. We show that only languages in BPP have constant-round Arthur–Merlin interactive proofs which are *black-box simulation zero-knowledge*. (For a definition of black-box simulation zero-knowledge and Arthur–Merlin interactive proofs, see §2.) We have the following theorem.

THEOREM 6.1. *A language L has a constant-round Arthur–Merlin interactive proof which is black-box simulation zero-knowledge if and only if $L \in BPP$.*

In §6.1 we present a proof for a special case of this theorem, namely, for the case of a three-round Arthur–Merlin protocol. The general case is proved in §6.2 using careful extensions of the ideas presented for this special case.

The three-round case can also be extended to general interactive proof systems. That is, we also have the following theorem, proved in §6.3.

THEOREM 6.2. *A language L has a three-round interactive proof which is black-box simulation zero-knowledge if and only if $L \in BPP$.*

(We remark that [GO] and [Ore] show that two-round (auxiliary-input) zero-knowledge proofs—not necessarily black-box simulation—exist only for BPP languages.)

Our results are optimal in the sense that there exist Arthur–Merlin interactive proofs, for languages believed to be outside BPP, with unbounded number of rounds and which are black-box simulation zero-knowledge. Similarly, there exist four-round interactive proof protocols (using private coins) which are also black-box simulation zero-knowledge. For further details about these protocols, and some consequences concerning the hierarchy of languages having zero-knowledge Arthur–Merlin proofs, see §1.

It is interesting to note that our results hold also for a weaker notion of black-box simulation zero-knowledge, namely, one which only requires the existence of a black-box simulator that succeeds in simulating conversations with *deterministic* (nonuniform) verifiers. The sufficiency of this condition follows from the proofs below. Also, the formulation of the completeness condition of an interactive proof (see §2) can be relaxed in the following way. We have defined the completeness condition by requiring that the prover convince the verifier of accepting an input in the language with probability almost 1 (i.e., 1 minus a negligible fraction). For the correctness of our results it suffices to require just a nonnegligible probability. (In this section we use this weaker formulation of the completeness condition.) On the other hand, the requirement of a negligible probability of convincing the verifier to accept an input not in the language (the soundness condition) is essential. (For example, three-round zero-knowledge protocols exist for all languages in NP if the soundness condition is formulated with probability $\frac{1}{2}$ [GMW1].) Finally, our results hold also in the setting of *interactive arguments* [BCC], i.e., "interactive proofs" in which the prover is limited to probabilistic polynomial-time computations, possibly getting an auxiliary input.

6.1. The case AM(3).

The protocol $\langle \mathbf{P}, \mathbf{V} \rangle$. Consider an Arthur–Merlin protocol $\langle P, V \rangle$ for a language L, consisting of three rounds. We use the following notation. Denote by x the input for the protocol, and by n the length of this input. The first message in the interaction is sent by the prover. We denote it by α. The second round is the V, which sends a string β. The third (and last) message is from P, and we denote it by γ. The predicate computed by the verifier V in order to accept or reject the input x is denoted by ρ_V, and we consider it, for convenience, as a deterministic function $\rho_V(x, \alpha, \beta, \gamma)$. (For the general case, see Remark 6.2.) We will also assume, without loss of generality, the existence of a polynomial $l(n)$ such that $|\alpha| = |\beta| = l(n)$.

The simulation process. Let this three-round Arthur–Merlin protocol $\langle P, V \rangle$ be black-box simulation zero-knowledge. Denote by M the guaranteed probabilistic expected polynomial-time black-box simulator which, given access to the black-box V^*, simulates $\langle P, V^* \rangle$. The process of simulation consists of several "tries" or calls to the interacting verifier V^* ("the black box"). In each such call the simulator M feeds the arguments for V^*. These arguments are the input y (which may be different from the "true" input x), the random coins for V^*, and a string α representing the message sent by the prover P. In our case, it suffices for our results to consider a simulator that is just able to simulate conversations with *deterministic* (*nonuniform*) *verifiers*. In particular, this simulator does not care about feeding the black-box V^* with random coins. This simplifies our proof by avoiding any reference to these random coins for V^*, and strengthen our result (since it holds even under the sole existence of this weak kind of simulator).

After completing its tries the simulator outputs a conversation $(y, \alpha, \beta, \gamma)$.

We shall make some further simplifying assumptions on the behavior of the simulator M, which will not restrict the generality. In particular, we assume that some cases, which may arise with only negligible probability, do not happen at all. This cannot significantly effect the success probability of the simulator. In other words, any black-box simulator which successfully simulates $\langle P, V^* \rangle$ conversations of deterministic verifiers V^* can be changed into another simulator for which the following conventions hold and which has the same success probability as the original simulator, except for a possibly negligible difference. We assume the following:

- The conversations output by M always have the form $(x, \alpha, \beta, \gamma)$ (i.e., $y = x$), and that the string β equals the message output by V^* when fed with inputs x and α. Note that these conditions always hold for the real conversations generated by the prover P and the (deterministic) verifier V^*. Therefore, the simulator must almost always do the same. (Otherwise, a distinguisher which has access to V^* would distinguish between the simulator's output and the original conversations.)
- The simulator M explicitly tries, in one of its calls to V^*, the arguments x and α appearing in the output conversation. (For example, once the simulator decides on the output conversation with a specific parameter α, it explicitly feeds V^* with x and this value of α, regardless of whether it asked α before or not. In any case, the answer of the deterministic V^* to the pair (x, α), will be always the same.)
- The simulator runs in (strictly) polynomial time. (In Remark 6.1 below we show how to handle the general case in which the simulator runs in *expected* polynomial time.) We denote by $t(n)$ a polynomial bounding the number of calls tried by M before outputting a conversation.

The simulator as a subroutine. Our goal is to present a BPP algorithm for the language L. The idea is to use the simulator M in order to distinguish between inputs in and outside L. For that, we use the simulator itself as a subroutine of the BPP algorithm. We do not make any assumption on the internal behavior of this simulator, but just use the following observation. *The behavior of the simulator M, interacting with a verifier V^*, is completely determined by the input x, the random tape R_M used by M, and the strings output by V^** (*in response to the arguments fed by the simulator during its tries*). Therefore, in order to operate M, we just need to feed it with an input x, a tape of random coins, and a sequence of responses to its messages α. Below we formally describe a computation process that uses M as a subroutine. (We stress that in this process there is no *explicit* verifier present.)

Fix an input x of length n, a string R_M (of length $q(n)$, where $q(\cdot)$ is a polynomial bounding the number of random coins used by M on inputs of length n), and $t = t(n)$ (arbitrary) strings $\beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(t)}$, each of length $l(n)$. Activate M on input x with its random tape containing R_M. For each y and α tried by M, respond with a message β from the above list $\beta^{(1)}, \beta^{(2)}, \ldots, \beta^{(t)}$ according to the following rule. (This rule depends on the strings α but not on y.) To the first α presented by M respond with $\beta^{(1)}$. For subsequent α's check whether the same string α was presented by M. If so, respond with the same β as in that case; if it is the first time this α is presented then respond with the first unused $\beta^{(i)}$ in the list. That is, if α is the ith *different* string presented by M then we respond with $\beta^{(i)}$. We denote the ith different α by $\alpha^{(i)}$. Clearly, $\alpha^{(i)}$ is uniquely determined by x, R_M, and the $i-1$ strings $\beta^{(1)}, \ldots, \beta^{(i-1)}$, i.e., there exists a deterministic function α_M such that $\alpha^{(i)} = \alpha_M(x, R_M, \beta^{(1)}, \ldots \beta^{(i-1)})$. We denote by $\mathrm{conv}_M(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)}) = (x, \alpha, \beta, \gamma)$ the conversation output by the simulator M when activated with these parameters (notice that t strings $\beta^{(i)}$ always suffice for answering all tries of M). By our convention on the simulator M, there exists i, $1 \le i \le t$, such that $\alpha = \alpha^{(i)}$ and $\beta = \beta^{(i)}$.

DEFINITION. *We say that a vector* $(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ *is M-good if* $\mathrm{conv}_M(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ *is an accepting conversation for the (honest) verifier V, namely, if* $\mathrm{conv}M(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)}) = (x, \alpha, \beta, \gamma)$ *and* $\rho_V(x, \alpha, \beta, \gamma) = ACCEPT$. *We say that* $(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ *is* (M, i)*-good (or i-good for short) if it is M-good and* $\alpha = \alpha^{(i)}$, $\beta = \beta^{(i)}$.

The main property of M-good strings is stated in the following lemma.

LEMMA 6.3. *Let* $\langle P, V\rangle$ *be a three-round Arthur–Merlin protocol for a language L. Suppose* $\langle P, V\rangle$ *is black-box simulation zero-knowledge, and let M be a black-box simulator as above. Then,*

1. *for strings x outside L, only a negligible portion of the vectors* $(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ *are M-good;*
2. *for strings x in L there exists a nonnegligible portion of the vectors* $(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ *that are M-good. (This nonnegligible portion is at least one half of the completeness probability of the protocol* $\langle P, V\rangle$*, i.e., at least half the probability that P convinces V to accept x.)*

Before proving this key lemma, we use it to prove Theorem 6.1 for the case of the three-round Arthur–Merlin interactive proof.

Proof of Theorem 6.1 (*for the case* AM(3)). By Lemma 6.3 we get the following BPP algorithm for the language L. On input x:

*select at random a vector $(R_M, \beta^{(1)}, \ldots, \beta^{(t)})$;

*accept x if and only if $(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ is M-good.

The complexity of this algorithm is like the complexity of testing for M-goodness. The latter is polynomial-time since it involves running the simulator M which is polynomial-time, and evaluating the predicate ρ_V, which is also polynomial-time computable. The success probability of the algorithm is given by Lemma 6.3. □

Proof of Lemma 6.3. (1) Assume that the portion of M-good vectors $(x, R, \beta^{(1)}, \ldots, \beta^{(t)})$ for x's not in L is not negligible. This means that there exist infinitely many $x \notin L$ for which the portion of M-good vectors is nonnegligible. For each such x, there exists an index i_0, $1 \le i_0 \le t$, for which a nonnegligible fraction of the vectors $(x, R, \beta^{(1)}, \ldots, \beta^{(t)})$ are i_0-good (since there are only polynomially many possible values for i_0). Thus, there exists a nonnegligible number of prefixes $(x, R, \beta^{(1)}, \ldots, \beta^{(i_0-1)})$, each with a nonnegligible number of i_0-good continuations $(\beta^{(i_0)}, \ldots, \beta^{(t)})$ (i.e., such that $(x, R, \beta^{(1)}, \ldots, \beta^{(i_0-1)}, \beta^{(i_0)}, \ldots, \beta^{(t)})$ are i_0-good). Let $(x, R, \beta^{(1)}, \ldots, \beta^{(i_0-1)})$ be such a prefix, and let $\alpha^{(i_0)} = \alpha_M(x, R, \beta^{(1)}, \ldots, \beta^{(i_0-1)})$. For each i_0-good continuation $(\beta^{(i_0)}, \ldots, \beta^{(t)})$ machine M outputs a conversation $(x, \alpha^{(i_0)}, \beta^{(i_0)}, \gamma)$ for which $\rho_V(x, \alpha^{(i_0)}, \beta^{(i_0)}, \gamma) = ACCEPT$. In particular, there exists a nonnegligible number of $\beta^{(i_0)}$ for which this happens.

In other words, for each x as above, there exists a string $\alpha_x (= \alpha^{(i_0)})$ for which the set $B(x, \alpha_x) = \{\beta : \exists\gamma, \rho_V(x, \alpha_x, \beta, \gamma) = ACCEPT\}$ is of nonnegligible size among all possible strings β. Consider now a ("cheating") prover that sends this α_x as its first message. If V responds with $\beta \in B(x, \alpha_x)$, the prover sends the corresponding γ, which convinces V to accept. Since V selects its messages β at random, then the probability of being convinced by the above prover is (at least) as big as the relative size of $B(x, \alpha_x)$, i.e., nonnegligible. Concluding, we have shown the existence of a prover that for infinitely many x's outside L convinces V to accept with nonnegligible probability. This contradicts the soundness condition of the protocol $\langle P, V\rangle$, and this part of the lemma follows.

(2) We show that for strings x in L a nonnegligible portion of the vectors $(x, R_M, \beta^{(1)}, \dots, \beta^{(t)})$ are M-good. We do it by considering the behavior of the simulator M when receiving "random-like" responses from the verifier. This behavior is analyzed by introducing a particular family of "cheating" verifiers, each of them associated to a different hash function from a family of *$t(n)$-wise independent hash functions*. The $t(n)$-wise independence (where $t(n)$ is the bound on the number of simulator's tries) achieves the necessary randomness from the verifiers' responses.

Let $x \in L$ and let n denote its length. Consider a family of hash functions H_n which map $l(n)$-bit strings into $l(n)$-bit strings, such that the locations assigned to the strings by a randomly selected hash function are uniformly distributed and $t(n)$-wise independent. (Recall that $l(n)$ is the length of messages α and β in the Arthur–Merlin protocol $\langle P, V\rangle$ for L, while $t(n)$ is the bound on the number of M's tries.) For properties and implementation of such functions, see, e.g., [Jof], [WC], and [CG]; in particular, we observe that such functions can be described by a string of length $t(n) \cdot l(n)$, i.e., polynomial in n.

For each hash function $h \in H_n$ we associate a (deterministic nonuniform) verifier V_h^*, which responds to the prover's message α with the string $\beta = h(\alpha)$ (V_h^* has wired in the description of h). Consider the simulation of $\langle P, V_h^*\rangle$ conversations by the simulator M. Fixing an input x, a random tape R_M for M, and a function $h \in H_n$, the whole simulation is determined. In particular, this (uniquely) defines a sequence of α's tried by the simulator, and the corresponding responses β of V_h^*. We denote by $\alpha^{(1)}, \alpha^{(2)}, \dots, \alpha^{(s)}$, the *different* values of α in these tries. When $s < t$, we complete this sequence to $\alpha^{(1)}, \dots, \alpha^{(s)}, \alpha^{(s+1)}, \dots, \alpha^{(t)}$, by adding $t - s$ strings α in some canonical way, such that the resultant $\alpha^{(1)}, \dots, \alpha^{(t)}$ are all different. Let $\beta^{(i)} = h(\alpha^{(i)})$, $1 \le i \le t$, and define $v(x, R_M, h) = (x, R_M, \beta^{(1)}, \dots, \beta^{(t)})$. Part (2) of the lemma follows from the following two claims.

CLAIM 1. For $x \in L$, there is a nonnegligible portion of the pairs (R_M, h) for which the vector $v(x, R_M, h)$ is M-good.

Proof. For any input x to the protocol $\langle P, V\rangle$, let p_x denote the probability that the prover P convinces V (the honest verifier) to accept x. In other words, p_x is the probability, over the coin sequences R_P of the prover P, and (random) choices β of V, that the resultant conversation $(x, \alpha(x, R_P), \beta, \gamma(x, R_P, \beta))$ is accepting. By the completeness property of the protocol $\langle P, V\rangle$, we get that for x's in L the probabilities p_x are nonnegligible.

Let $x \in L$ and consider the interaction between the real prover P and the verifiers V_h^* on the input x. Each coin sequence R_P determines the message α and the corresponding response $h(\alpha)$ by V_h^*. By the uniformity property of the family H_n we get that for every α, all β's are equiprobable as the result of $h(\alpha)$. Therefore, the probability that P and V_h^* (for h uniformly chosen from H_n) output an accepting conversation is exactly the same as the probability, p_x, that P and V output such a conservation.

Finally, since the simulator M succeeds in simulating $\langle P, V_h^*\rangle$ conversations for all functions $h \in H_n$, we get that for each h the probability that M outputs an accepting conversation when interacting with V_h^* is almost the same (up to negligible difference) as the probability that P and V_h^* output an accepting conversation. This last probability, for $h \in_R H_n$, is p_x. We conclude that the probability, over random R_M and h, that $v(x, R_M, h)$ is M-good is almost p_x and thus nonnegligible. The claim follows. □

CLAIM 2. For all strings x and R_M, and for h chosen with uniform probability from H_n, the vector $v(x, R_M, h)$ is uniformly distributed over the set $\{(x, R_M, \beta^{(1)}, \dots, \beta^{(t)}) : \beta^{(i)} \in \{0,1\}^{l(n)}\}$.

Proof. Recall the function α_M introduced above. Observe that

$$v(x, R_M, h) = (x, R_M, \beta^{(1)}, \dots, \beta^{(t)})$$

if and only if for every i, $1 \le i \le t$,

$$h(\alpha_M(x, R_M, \beta^{(1)}, \dots, \beta^{(i-1)})) = \beta^{(i)}.$$

On the other hand, by the uniformity and $t(n)$-independence property of the family H_n, we have that for any t *different* elements $a_1, \dots, a_t$ in the domain of the functions $h \in H_n$, the sequence $h(a_1), \dots, h(a_t)$ is uniformly distributed over all the possible sequences $b_1, \dots, b_t$ for b_i in the range of the functions H_n.

Thus, for all strings x and R_M, and for fixed $\beta^{(1)}, \dots, \beta^{(t)}$, the probability (for $h \in_R H_n$) that $v(x, R_M, h) = (x, R_M, \beta^{(1)}, \dots, \beta^{(t)})$ equals the probability that for every i, $1 \le i \le t$, h maps $\alpha^{(i)} = \alpha_M(x, R_M, \beta^{(1)}, \dots, \beta^{(i-1)})$ into $\beta^{(i)}$. Since, by definition, all $\alpha^{(i)}$'s are different, then we can use the above property of the family H_n to get that the latter probability is the same for every sequence $\beta^{(1)}, \dots, \beta^{(t)}$ (i.e., we put $a_i = \alpha^{(i)}$ and $b_i = \beta^{(i)}$). The claim follows. □

Claim 2 states that for any R_M, the value of $v(x, R_M, h)$ is uniformly distributed over all possible vectors $(x, R_M, \beta^{(1)}, \dots, \beta^{(t)})$. On the other hand, by Claim 1, a nonnegligible portion of $v(x, R_M, h)$ are M-good, and then we get that a nonnegligible portion of the vectors $(x, R_M, \beta^{(1)}, \dots, \beta^{(t)})$ are M-good.

The lemma follows. □

Remark 6.1 (*expected polynomial-time simulator*). For simplicity we have assumed that the given simulator, M, for the protocol $\langle P, V \rangle$ runs in (strictly) polynomial time. Nevertheless, in the definition of zero-knowledge we allow this simulator to run in *expected* polynomial time. We show that our results also hold in this general case by transforming a given expected polynomial-time simulator M into a strictly polynomial-time simulator M', and showing that Lemma 6.3 holds for this new simulator. Then, we can use the modified simulator M' in the BPP algorithm for the language L.

The simulator M' behaves like M, but its running time is truncated after some (fixed) polynomial number of steps, denoted $s(n)$. We show how to choose this polynomial $s(n)$. Let $T(n)$ be a polynomial bounding the *expected* running time of M, and let $p(n)$ be a (lower) bound on the probability that the prover P convinces the (honest) verifier V to accept an input in L of length n. We define $s(n)$ to be $2 \cdot T(n)/p(n)$. Since $1/p(n)$ is polynomially bounded (by the completeness condition of the protocol $\langle P, V \rangle$), then $s(n)$ is polynomially bounded. With this modification of M the proof of Lemma 6.3 remains valid, except for a more delicate argument in the proof of Claim 1. The required changes follow.

In that proof we claimed that "for each h the probability that M outputs an accepting conversation when interacting with V_h^* is almost the same (up to a negligible difference) as the probability that P and V_h^* output an accepting conversation." This is true for the original simulator M, but not necessarily for M'. Since we cut the running of M after $s(n)$ steps, then there exist cases in which M' does not complete the original behavior of M. Nevertheless, by the choice of $s(n)$, the probability (over the coin tosses of M') that this happens (i.e., the running time of M exceeds $s(n)$) is at most $p(n)/2$. Thus, for any h, the probability that the truncated simulator, M', outputs an accepting conversation when interacting with V_h^* differs from the probability that P and V_h^* output an accepting conversation by at most $p(n)/2$. For $h \in_R H_n$, this last probability was shown (in the original proof of Claim 1) to be at least $p(n)$, and then we get that the probability, over random $R_{M'}$ and h, that $v(x, R_{M'}, h)$ is M'-good is (up to a negligible difference) larger than $p(n)/2$, and then nonnegligible. Therefore, Claim 1 follows in this case also. □

Remark 6.2 (*randomized* ρ_V). We have assumed that the only coin tosses of the (honest) verifier V during the Arthur–Merlin protocol $\langle P, V \rangle$ are the bits corresponding to the string β sent to the prover, and that no additional coin tosses are used in order to compute the accepting/rejecting predicate ρ_V. This restriction can be removed from the above proof by using finer arguments, as done in our treatment of the general IP(3) case (of §6.3).

More generally, any AM(k) protocol in which the predicate ρ_V depends on the whole conversation and some additional random string can be transformed into an AM($k + 1$) protocol in which no such additional string is used: simply let the verifier send this random string as its last message. Hence, since we prove our result for any constant-round AM protocol, we can assume that ρ_V is deterministic. □

Remark 6.3 (*interactive arguments*). We now show how to generalize the above proof of the case AM(3) in order to prove the same result in the setting of interactive arguments, i.e., "interactive proofs" in which the soundness condition is required only with respect to provers limited to probabilistic polynomial-time computations, possibly getting an auxiliary input. We have to prove Lemma 6.3 in this setting. Notice that part (2) of the lemma relies on the completeness and zero-knowledge properties of the interactive proof, but these properties are not influenced by the soundness condition. Therefore, this part of the proof automatically holds for interactive arguments. The other part, part (1), relies on the soundness of the interactive proof, thus a modification is required in the proof to deal with provers having just polynomial power.

In that proof we showed, by contradiction, the existence of infinitely many x's not in L for which a cheating prover can convince the verifier to accept x with nonnegligible probability. The success of this prover was shown by proving, for each such x, the *existence* of a message α_x that for nonnegligibly many β's a string γ exists such that $\rho_V(x, \alpha_x, \beta, \gamma) = ACCEPT$. In the interactive arguments, setting the sole existence of such an α_x is not sufficient. The limited prover should find in probabilistic polynomial time this string and the corresponding response γ to the message β sent by V. We describe such a prover P^*, which uses the simulator M in order to find the required strings. It begins by choosing $i \in_R \{1, \dots, t\}$ and random strings R_M, $\beta^{(1)}, \dots, \beta^{(i-1)}$. Then it computes $\alpha = \alpha_M(x, R_M, \beta^{(1)}, \dots, \beta^{(i-1)})$ and sends this α to V. Once V responds with β, the prover P^* chooses $t-i$ random strings $\beta^{(i+1)}, \dots, \beta^{(t)}$, computes (using the simulator M) the conversation $\text{conv}_M(x, R_M, \beta^{(1)}, \dots, \beta^{(i-1)}, \beta, \beta^{(i+1)}, \dots, \beta^{(t)})$, and sends to V the message γ appearing in this conversation. If the chosen vector is i-good then this γ convinces V to accept the conversation. We analyze the probability of such an event.

There exists a nonnegligible probability that P^* chooses i, $1 \leq i \leq t$, for which the number of i-good vectors is nonnegligible (we saw that such an i exists). On the other hand, the whole vector $(x, R_M, \beta^{(1)}, \dots, \beta^{(i-1)}, \dots, \beta^{(t)})$ is chosen at random (except for x): the β component by the verifier (the protocol is Arthur–Merlin!) and the other components by P^*. Therefore, there is a nonnegligible probability that the resultant vector is i-good, in which case V accepts x. This way P^* works in polynomial time and has a nonnegligible probability of convincing V to accept x, from which we derive the required contradiction. □

6.2. The case AM(k): Secret coins help zero-knowledge.

In this section we consider constant-round Arthur–Merlin interactive proofs. We show that a language having such an interactive proof which is also black-box simulation zero-knowledge belongs to BPP, thus proving Theorem 6.1. We present this proof based on the proof for the particular case of AM(3) as given in §6.1. The basic ideas are similar, but their implementation is technically more involved in this general case. We highly recommend familiarity with §6.1 before going through the present section.

The protocol ⟨P, V⟩. Let $\langle P, V \rangle$ be a k-round Arthur–Merlin protocol for a language L. For simplicity of the exposition we make some assumptions on the form of the protocol without restricting the generality of the proof. We consider protocols in which both the first and last messages are sent by the prover. By adding dummy messages any protocol can be converted into one of this form. Notice that in such a protocol, the number of rounds is always an odd number $k = 2 \cdot m + 1$. The prover P sends $m + 1$ messages which we denote by $\alpha_1, \dots, \alpha_m$ and γ, respectively. The m messages by V are denoted $\beta_1, \dots, \beta_m$. The input to the protocol is denoted by x, and its length by n. The predicate computed by the verifier V in order to accept or reject the input x is denoted by ρ_V, and we assume it to be a deterministic function of the conversation $\rho_V(x, \alpha_1, \beta_1, \dots, \alpha_m, \beta_m, \gamma)$. (Our results hold also for interactive proofs in which ρ_V depends on an additional random string. See Remark 6.2.) We need the following technical convention. We assume that all prover messages in the protocol have a form that allows them, by only seeing the ith message α_i, to uniquely reconstruct all previous messages sent by the prover during the conversation. This is easily achieved by simple concatenation of previous messages (using a delimiter or some length convention). We also assume the existence of a polynomial $l(n)$ such that all prover's and verifier's messages on an n-length input have length $l(n)$ (e.g., using dummy padding). Finally, we let the verifier V check whether the received messages conform to the above conventions, and reject the conversation if not.

The simulation process. We denote by M the black-box simulator for the protocol $\langle P, V\rangle$. The simulation process consists of several tries by the simulator M. Each try involves feeding the verifier V^* (i.e., the black box representing it) with a value y as the input to the protocol, and the messages α_i, $1 \le i \le m$, that simulate the messages sent by P. (Again, we do not care about random coins for V^*; we just need a simulator that is able to simulate conversations with deterministic verifiers.) The simulator M chooses these arguments, in the successive tries, depending on the random tape R_M and the responses β_i output by the black box V^* during the current and previous tries. After each try the simulator may decide to output a conversation of the form $(y, \alpha_1, \beta_1, \dots, \alpha_m, \beta_m, \gamma)$ or to perform a new try. We assume that the output conversation has $y = x$ (i.e., the input component in the conversation corresponds to the actual input being simulated), that the α messages appearing in the output conversation fit our convention on the form of the prover's messages, and that the simulator explicitly tries the output conversation. Namely, it operates (in one of the tries) the black box V^* on input x and $\alpha_1, \dots, \alpha_m$ as appearing in the output conversation, and, respectively, gets as responses to V^* the strings $\beta_1, \dots, \beta_m$, also appearing in this conversation. These assumptions are apparently restricting ones, since the simulator is allowed to output conversations that are not "legal conversations" between the prover P and the simulated verifier V^*. Nevertheless, a simulator that succeeds simulating the $\langle P, V^*\rangle$ conversations will output such illegal conversations with only negligible probability (otherwise the simulated conversations can be easily distinguished from the true ones). Finally, we consider, for the sake of simplicity, only simulators that run in (strictly) polynomial time. The necessary changes in the proof for handling the general case in which the simulator runs in expected polynomial time are analogous to the ones described in Remark 6.1 for the case AM(3). We denote by $\hat{t}(n)$ a polynomial bounding the number of calls to V^* tried by M before outputting a conversation, and put $t(n) = m \cdot \hat{t}(n)$ (notice that $t(n)$ constitutes an upper bound on the total number of messages α tried by M during the whole simulation).

The simulator as a subroutine. Our goal is to present a BPP algorithm for the language L, and we use the simulator M to achieve it. The way M is used is similar to the way we used the simulator in the AM(3) case (see §6.1). In the present case, the behavior of the simulator M when "interacting" with a verifier V^* is determined by the input x to the protocol, the random tape R_M, and the strings β output by V^* as responses to the strings fed by M during the different tries. Also, here we define a computational process that uses M as a subroutine.

Fix an input x of length n, a string R_M, and $t = t(n)$ strings $\beta^{(1)}, \beta^{(2)}, \dots, \beta^{(t)}$, each of length $l(n)$. Activate M on input x with its random tape containing R_M. For each message α presented by M, respond in the following way. (The responses will depend on the strings α, but not on y.) If α is "illegal," then respond with a special "reject-message." By illegal we mean a message α that does not fit our above conventions on the form of the prover's messages. For legal α's we respond (impersonating a black-box verifier) with one of the β's from the above list $\beta^{(1)}, \dots, \beta^{(t)}$ according to the following rule. If the same α was previously presented by M (i.e., during a previous try), respond with the same β as in that case. If α is the ith *different* (legal) string presented by M since the beginning of the simulation, then respond with $\beta^{(i)}$. We denote the ith different α by $\alpha^{(i)}$. Clearly, $\alpha^{(i)}$ is uniquely determined by x, R_M, and the $i-1$ strings $\beta^{(1)}, \ldots, \beta^{(i-1)}$. That is, there exists a deterministic function α_M such that $\alpha^{(i)} = \alpha_M(x, R_M, \beta^{(1)}, \ldots, \beta^{(i-1)})$. We denote by $\text{conv}_M(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)}) = (y, \alpha_1, \beta_1, \ldots, \alpha_m, \beta_m, \gamma)$ the conversation output by the simulator M when activated with these parameters (notice that t strings $\beta^{(i)}$ always suffice for answering all tries of M). By our convention on the simulator M and on the form of the prover's messages it follows that there exists a sequence of indices $1 \leq i_1 < i_2 < \cdots < i_m \leq t$ such that for each α_j, β_j, $j = 1, \ldots, m$, appearing in the output conversation, $\alpha_j = \alpha^{(i_j)}$ and $\beta_j = \beta^{(i_j)}$. This is true since the simulator always outputs a conversation which was explicitly generated in one of its tries. The increasing property of the sequence of indices i_j is enforced by the special form of the "legal" messages α, namely, by the fact that we respond to message α_j only if we had previously responded to the messages $\alpha_1, \ldots, \alpha_{j-1}$. In the present setting we use the following definition of M-good vectors.

DEFINITION. *We say that a vector* $(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ *is M-good if* $\text{conv}_M(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ *is an accepting conversation for the (honest) verifier V. We say that* $(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ *is* $(i_1, i_2, \ldots, i_m)$*-good if it is M-good and the corresponding conversation has* $\alpha_j = \alpha^{(i_j)}$ *and* $\beta_1 = \beta^{(i_j)}$*, for* $j = 1, \ldots, m$.

The following lemma is analogous to Lemma 6.3.

LEMMA 6.4. *Let* $k = 2 \cdot m + 1$ *be a constant, and let* $\langle P, V \rangle$ *be a k-round Arthur–Merlin protocol for a language L. Suppose* $\langle P, V \rangle$ *is black-box simulation zero-knowledge, and let M be a black-box simulator as above. Then*

1. *for strings x outside L, only a negligible portion of the vectors* $(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ *are M-good*;
2. *for strings x in L there exists a nonnegligible portion of the vectors* $(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ *that are M-good. (This nonnegligible portion is at least one half of the completeness probability of the protocol* $\langle P, V \rangle$*, i.e., half the probability that P convinces V to accept x).*

Proof of Theorem 6.1. Using Lemma 6.4 we get that the algorithm described in the proof of Theorem 6.1 for the special case of AM(3) (see §6.1) is a BPP algorithm for the language L. □

Proof of Lemma 6.4. This proof is essentially analogous to the proof of Lemma 6.3, although some delicate modifications are required.

(1) Assume that the portion of M-good vectors $(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ for x's not in L is not negligible. This means that there exist infinitely many $x \notin L$ for which the portion of M-good vectors is nonnegligible. Observe that there are only polynomially many different sequences $1 \leq i_1 < i_2 < \cdots < i_m \leq t$ (i.e., $\binom{t(n)}{m}$, and m is a constant), and then note that for each x, as above, there exists a sequence $(i_1, i_2, \ldots, i_m)$ for which nonnegligibly many vectors $(x, R_M, \beta^{(1)}, \ldots, \beta^{(t)})$ are $(i_1, i_2, \ldots, i_m)$-good. Next, we describe a prover P^* which convinces the (honest) verifier V to accept any of the above inputs $x \notin L$ with nonnegligible probability, thus contradicting the soundness condition of the protocol $\langle P, V \rangle$.

The prover P^* begins by choosing a sequence $(i_1, i_2, \ldots, i_m)$ at random. Then, it chooses random strings $R_M, \beta^{(1)}, \ldots, \beta^{(i_1-1)}$ and uses them to compute $\alpha_1 = \alpha_M(x, R_M, \beta^{(1)}, \ldots, \beta^{(i_1-1)})$. It sends α_1 to V and receives back the response β_1. Now P^* chooses random $\beta^{(i_1-1)}, \ldots, \beta^{(i_2-1)}$ and computes $\alpha_2 = \alpha_M(x, R_M, \beta^{(1)}, \ldots, \beta^{(i_1-1)}, \beta_1, \beta^{(i_1+1)}, \ldots, \beta^{(i_2-1)})$. After receiving the response β_2 from the verifier, P^* selects new random strings $\beta^{(i_2+1)}, \ldots, \beta^{(i_3-1)}$ and computes $\alpha_3 = \alpha_M(x, R_M, \beta^{(1)}, \ldots, \beta^{(i_1-1)}, \beta_1, \beta^{(i_1+1)}, \ldots, \beta^{(i_2-1)}, \beta_2, \beta^{(i_2+1)}, \ldots, \beta^{(i_3-1)})$. This process continues until all messages α_i, β_i, $1 \leq i \leq m$, are computed and exchanged. When the resultant vector $(x, R_M, \beta^{(1)}, \ldots, \beta^{(i_1-1)}, \beta_1, \beta^{(i_1+1)}, \ldots,$

$\beta^{(i_2-1)}, \beta_2, \ldots, \beta^{(t)})$ is $(i_1, i_2, \ldots, i_m)$-good, then computing the function conv_M on this vector results in an accepting (for V) conversation $(x, \alpha_1, \beta_1, \ldots, \alpha_m, \beta_m, \gamma)$ (with α_i, β_i, as defined above). But then, by sending this γ, the prover P^* convinces V to accept. The probability that this happens equals the probability that the above vector $(x, R_M, \beta^{(1)}, \ldots, \beta^{(i_1-1)}, \beta_1, \beta^{(i_1+1)}, \ldots, \beta^{(i_2-1)}, \beta_2, \ldots, \beta^{(t)})$ is $(i_1, i_2, \ldots, i_m)$-good. Since this sequence of indices and all the vector components (excluding x) are chosen at random (recall that V chooses its messages, $\beta_1, \ldots, \beta_m$, at random!) then this probability is nonnegligible.

(2) The proof of this part is analogous to the corresponding proof in Lemma 6.3. We use a set H_n of $t(n)$-independent hash functions ($t(n)$ as defined in this section) to define a family of verifiers V_h^*. For all $h \in H_n$, the verifier V_h^* responds to a legal message α sent by the prover with $h(\alpha)$, and with a rejection message if α is illegal. The statements for Claims 1 and 2 remain the same, as does the proof of Claim 2. The proof of Claim 1 needs a more delicate argument, as follows. As in the AM(3) case we consider the interaction between the prover P and a verifier V_h^*, but now this interaction generates a conversation of the form $(x, \alpha_1, \beta_1, \ldots, \alpha_m, \beta_m, \gamma)$. In particular, for each h and random tape R_P for P a unique sequence of messages $\beta_1, \ldots, \beta_m$ (the responses of V_h^*) is determined. We have to show that for every tape R_P all sequences $\beta_1, \ldots, \beta_m$ are equiprobable for $h \in_R H_n$. The proof of this property uses a similar argument as in the proof of Claim 2: observe that the pair R_P and h generates the responses $\beta_1, \ldots, \beta_m$ if and only if, for every i, $1 \le i \le m$, $h(\alpha_P(x, R_P, \beta_1, \ldots, \beta_{i-1})) = \beta_i$. (Here α_P stands for the function computed by P in order to determine its next message α.) Thus, the probability (for $h \in_R H_n$) that a given sequence $\beta_1, \ldots, \beta_m$ is generated is like the probability that for every i, $1 \le i \le m$, h maps $\alpha_P(x, R_P, \beta_1, \ldots, \beta_{i-1})$ into β_i. Since the functions H_n are m-independent (by definition they are $t(n)$-independent, but $m \le t(n)$), and the messages $\alpha_1, \ldots \alpha_m$ output by P are all different by convention, we get that the latter probability is the same for every sequence $\beta_1, \ldots, \beta_m$.

From this property of the pairs R_P and h we conclude that the probability that P and V_h^* (for $h \in_R H_n$) output an accepting conversation is exactly the same as the probability that P and the honest V output such a conversation.

The rest of the proof follows as in Claim 1 of Lemma 6.3 □

Remark 6.4. Notice that the prover P^* described in the proof of part (1) of Lemma 6.4 is a polynomial-time prover. The other parts of the proof of Theorem 6.1 also hold for such provers, and then we get that our result remains valid also in the setting of interactive arguments.

6.3. The case IP(3). In the setting of general interactive proofs the (honest) verifier is not restricted to choosing all its messages at random, but can compute them based on the input x, a random ("secret") string r, and the previous messages of the prover. In the case of three rounds this means that the only message sent by V during the protocol is computed by means of a (deterministic) function $\beta_V(x, r, \alpha)$, where α is the first message sent by P. Also, in this case V accepts or rejects a conversation based on a predicate $\rho_V(x, r, \alpha, \gamma)$ (γ is the last message sent by P).

Here we outline the proof of Theorem 6.2, based on the proof presented in §6.1 for the AM(3) case.

Proof of Theorem 6.2 (*outline*). Let L be a language having a three-round interactive proof which is black-box simulation zero-knowledge. Let $\langle P, V\rangle$ be such a protocol and let M be the corresponding black-box simulator. The simulation process consists of several tries; in each of them the simulator feeds the black box V^* with arguments y (the input) and α (the prover's message), and gets an answer β from V^*. (Again, it suffices to consider a simulator just able to simulate conversations with deterministic verifiers, so this simulator does not feed V^* with a random tape.) We assume the same conventions on the simulator as the ones described in §6.1 for the proof of the AM(3) case.

- The simulator always outputs a conversation of the form $(x, \alpha, \beta, \gamma)$, where x and α are fed into the black-box V^* in one of the simulator tries, and β is the response of V^* to these arguments.
- The simulator runs in strictly polynomial time. In particular, $t(n)$ stands for the polynomial bound on the number of tries made by M on inputs of length n during the simulation process (the case of expected polynomial-time simulators is handled exactly as in Remark 6.1).

The main modification with respect to the proof of the AM(3) case is in the way we use the simulator M as a subroutine for constructing the BPP algorithm for the language L. Recall that the whole simulator process is completely determined by the input to the protocol, x, the contents of M's random tape, R_M, and the responses by the verifier. This was true for the AM(3) case and remains true here. In the former case we used M as a subroutine by feeding it with x and a randomly chosen string R_M. Then, we used $t = t(n)$ random strings $\beta^{(1)}, \ldots, \beta^{(t)}$ as the response of the virtual verifier. In the present case we choose a string R_M, as before, and t random strings denoted $r^{(1)}, \ldots, r^{(t)}$, each of length $l(n)$, where $l(n)$ is a (polynomial) bound on the number of random bits used by the (honest) verifier in the IP(3) protocol $\langle P, V \rangle$. The idea is to use these strings as the random coins of the virtual verifier for responding to M's tries. More precisely, for each try by the simulator, consisting of an input y to the protocol and a message α, we compute $\beta = \beta_V(y, r^{(i)}, \alpha)$, and feed it into M as the verifier's response to α. For each new try we use a new $r^{(i)}$ (in increasing order of i), except in the case in which the present α was also presented in a previous try. If so, we use the same $r^{(i)}$ as in that case.

Note that a unique conversation is determined by x, R_M, and the t strings $r^{(1)}, \ldots, r^{(t)}$. Thus, as in the case AM(3), we can define $\text{conv}_M(x, R_M, r^{(1)}, \ldots, r^{(t)})$ to be the conversation output by M when the described process is finished. Also, we denote by $\alpha^{(i)}$ the ith *different* α output by M during the simulation. Clearly, $\alpha^{(i)}$ is uniquely determined by x, R_M, and the strings $r^{(1)}, \ldots, r^{(i-1)}$; thus we denote $\alpha^{(i)} = \alpha_M(x, R_M, r^{(1)}, \ldots, r^{(i-1)})$.

By our convention on M, if $\text{conv}(x, R_M, r^{(1)}, \ldots, r^{(t)}) = (x, \alpha, \beta, \gamma)$, then M explicitly tried the arguments x and α during the simulation, and got as response the string β. This means that there exists (at least one) i, $1 \leq i \leq t$, such that $\beta = \beta_V(x, r^{(i)}, \alpha)$. This fact is used in the following definition.

DEFINITION. *We say that a vector* $(x, R_M, r^{(1)}, \ldots, r^{(t)})$ *is M-good if* $\text{conv}(x, R_M, r^{(1)}, \ldots, r^{(t)}) = (x, \alpha, \beta, \gamma)$ *and* $\rho_V(x, r^{(i)}, \alpha, \gamma) = ACCEPT$, *where i is the minimal value for which* $\beta = \beta_V(x, r^{(i)}, \alpha)$. *According to this value of i, we call the conversation (M, i)-good (or i-good for short).*

Using this re-definition of the notion of M-goodness, Lemma 6.3 of §6.1 also holds in the present (IP(3)) case, by just changing the $\beta^{(i)}$ notation by $r^{(i)}$ in the formulation of the lemma. Theorem 6.2 then follows by using the BPP algorithm as described in the proof of the AM(3) case. For the proof of Lemma 6.3 in the present case we note the following simple modifications. In the proof of part (1) we use the same reasoning as in the corresponding proof in §6.1 but applied to the strings $r^{(i)}$ instead of $\beta^{(i)}$. We note that the soundness probability of the protocol is now defined over the random coins used by V, i.e., over the choices $r^{(i)}$. For the proof of the other part of the lemma we slightly modify the definition of the verifiers V_h^*. We still use the same family of hash functions, but the verifier V_h^* works as follows: on message α sent by the prover, V_h^* responds with $\beta = \beta_V(x, h(\alpha), \alpha)$, i.e., it computes β as the honest verifier does, but using $h(\alpha)$ as the random coins of V. The rest of the proof (including Claims 1 and 2) remains essentially unchanged (up to the replacement of "responses $\beta^{(i)}$" by "random coins $r^{(i)}$"). □

Remark 6.5. As in the previous cases also, the IP(3) case extends to the setting of interactive arguments. The modifications in the proof are analogous to the ones described in Remark 6.3.

7. Concluding remarks.

Although the results presented in this paper are negative in nature, we believe that they have played a positive role in the development of the field.

We believe that sequential composition is a fundamental requirement of zero-knowledge protocols. It is analogous to requiring that adding two algebraic expressions, each evaluating to zero, yields an expression which evaluates to zero. Furthermore, sequential composition is required when using zero-knowledge proofs as tools in the design of cryptographic protocols (an application which is the primary motivation of zero-knowledge). *The fact that the original formulation of zero-knowledge is not closed under sequential composition establishes the importance of augmenting this formulation by an auxiliary input* (cf. [GO], [Ore], [TW], and [Gol]). It should be stressed, of course, that all known zero-knowledge proofs also satisfy the augmented formulation.

Parallel composition is the key to improving the efficiency (in terms of number of rounds) of zero-knowledge protocols, but we do not believe that it is a fundamental requirement. Carrying the analogy of the previous paragraph, one cannot require that "interleaving" two expressions (each evaluating to zero) yields an expression which evaluates to zero. *The fact that all known formulations of (computational) zero-knowledge are not closed under parallel composition motivates the introduction of weaker notions such as* witness indistinguishability [FS2] *which suffice for many applications.* Namely, instead of strengthening the hypothesis of the alleged Parallel Composition Theorem (as was done in the case of Sequential Composition), one relaxes the conclusion of the Parallel Composition Theorem (and this weaker conclusion turns to suffice in many applications).

The fact that ("nontrivial") black-box zero-knowledge proofs cannot be both of AM type and of constant number of rounds establishes the importance of "private coins" in the design of constant-round zero-knowledge proofs. In other words, in the process of such proofs, the verifier must "commit" (and later "decommit") to some pieces of information. In fact, such commitments are the core of the constant-round zero-knowledge proofs (and arguments) for any language in NP presented in [BCY], [FS1], and [GKa] (relying on various reasonable intractability assumptions) and in the (unconditional) zero-knowledge proof for graph isomorphism presented in [BMO1].

Appendix: Proof of existence of P/poly-evasive pseudorandom ensembles.

In this appendix we present the proof of Theorem 3.2. We first restate the theorem.

THEOREM 3.2. *There exists a* P/poly*-evasive ensemble* $\mathbf{S}^{(1)}, \mathbf{S}^{(2)}, \ldots$ *with* $Q(n) = 4n$*, such that for every* n*, each* $S_i^{(n)}$ *is a* $(2^{n/4}, 2^{-n/4})$*-pseudorandom set of cardinality* 2^n*. Furthermore, there exists a Turing machine which on input* 1^n *outputs the collection* $\mathbf{S}^{(n)}$*.*

Proof. For any integer n, we denote by $\mathbf{R}^{(n)}$ the collection of sets $S \subseteq \{0, 1\}^{4n}$ of cardinality 2^n which are $(2^{n/4}, 2^{-n/4})$-pseudorandom, and by $\mathbf{C}^{(n)}$ the set of (deterministic) circuits of size $2^{n/4}$ having n inputs and $4n$ outputs.

We prove the theorem by showing, for any large enough n, the existence of 2^n sets $S_1, \ldots, S_{2^n}$ from $\mathbf{R}^{(n)}$ such that for any circuit $C \in \mathbf{C}^{(n)}$, and $i \in_R \{1, \ldots, 2^n\}$, $\text{Prob}(C(i) \in S_i) < 2^{-n/4}$. Denoting this collection of 2^n sets by $\mathbf{S}^{(n)}$, we get that the resultant sequence $\mathbf{S}^{(1)}$, $\mathbf{S}^{(2)}, \ldots$ by a P/poly-evasive ensemble that satisfies the conditions of Theorem 3.2. We stress that considering only deterministic circuits does not restrict the generality, since we can wire in such a circuit a sequence of "random coins" that maximizes the probability $\text{Prob}(C(i) \in S_i)$.

We turn to show the existence of a collection of sets as described above. We do it by proving that there exists a positive probability to randomly choose 2^n sets $S_1, \ldots, S_{2^n}$ from $\mathbf{R}^{(n)}$ with the above evasivity property.

For a fixed $C \in \boldsymbol{C}^{(n)}$ and a fixed $i, 1 \leq i \leq 2^n$, consider the probability, denoted $\text{Prob}_S(C(i) \in S)$, that the element $C(i)$ belongs to the set S, for S uniformly chosen over all subsets of $\{0, 1\}^{4n}$ of size 2^n. Clearly,

$$\text{Prob}(C(i) \in S) = 1 - \frac{\binom{2^{4n}-1}{2^n}}{\binom{2^{4n}}{2^n}} = \frac{2^n}{2^{4n}} < \frac{1}{2^{2n}}.$$

We call a set $S \subseteq \{0, 1\}^{4n}$, $|S| = 2^n$, *C-bad* if there exists some $i, 1 \leq i \leq 2^n$, such that $C(i) \in S$. Fixing a circuit C, we have that for S uniformly chosen over all subsets of $\{0, 1\}^{4n}$ of size 2^n,

$$\text{Prob}_S(S \text{ is } C\text{-bad}) \leq \sum_{i=1}^{2^n} \text{Prob}_S(C(i) \in S) < 2^n 2^{-2n} = 2^{-n}.$$

In [GK] it is proven that the measure of $\boldsymbol{R}^{(n)}$ (i.e., the proportion of sets S which are $(2^{n/4}, 2^{-n/4})$-pseudorandom) is at least $1 - 2^{-2^{n/4}}$. Therefore, for each circuit $C \in \boldsymbol{C}^{(n)}$ the probability, hereafter denoted as p_C, to uniformly choose from $\boldsymbol{R}^{(n)}$ a set S which is C-bad is

$$p_C = \text{Prob}_S(S \text{ is } C\text{-bad} \mid S \in \boldsymbol{R}^{(n)}) < \frac{2^{-n}}{1 - 2^{-2^{n/4}}}.$$

We now proceed to compute the probability that for a fixed circuit $C \in \boldsymbol{C}^{(n)}$, a collection of 2^n randomly chosen sets from $\boldsymbol{R}^{(n)}$ contains a significant portion of C-bad sets. We define as "significant" a fraction $p_C + \delta_n$. (The quantity δ_n will be determined later.) Let ρ be a random variable assuming as its value the fraction of C-bad sets on a random sample of 2^n sets from $\boldsymbol{R}^{(n)}$. Clearly, the expected value of ρ is p_C. Using Hoeffding's inequality [Hoe] (see also [GK]) we get that

$$\text{Prob}(\rho \geq p_C + \delta_n) \leq e^{-2 2^n \delta_n^2},$$

i.e., this quantity bounds the probability of choosing at random 2^n sets from $\boldsymbol{R}^{(n)}$ among which the fraction of C-bad sets is larger than $p_C + \delta_n$.

Recall that we are interested in choosing 2^n sets that are evasive for *all* circuits $C \in \boldsymbol{C}^{(n)}$. That is, we require that for *any* C, the number of C-bad sets among the 2^n sets we choose is negligible. In order to bound the probability that 2^n randomly selected sets *do not* satisfy this condition, we multiply the above probability, computed for a single circuit, by the total number of circuits in $\boldsymbol{C}^{(n)}$ which is at most $2^{(2^{n/4})^2} = 2^{2^{n/2}}$. Putting $\delta_n = 2^{-n/4}/\sqrt{2}$ we get

$$2^{2^{n/2}} \cdot e^{-2 2^n \delta_n^2} = 2^{2^{n/2}} \cdot e^{-2 2^n 2^{-\frac{n}{2}-1}} = 2^{2^{n/2}} \cdot e^{-2^{n/2}} < 1.$$

We conclude that there exists a positive probability that 2^n sets $S_1, \ldots, S_{2^n}$ chosen at random from $\boldsymbol{R}^{(n)}$ have the property that for any circuit $C \in \boldsymbol{C}^{(n)}$ the fraction of C-bad sets among $S_1, \ldots, S_{2^n}$ is less than $p_C + \delta_n$. Therefore, such a collection of sets does exist.

Finally, we bound, for this fixed collection $S_1, \ldots, S_{2^n}$, and for any circuit $C \in \boldsymbol{C}^{(n)}$, the probability $\text{Prob}_i(C(i) \in S)i)$, for i randomly chosen from $\{1, \ldots, 2^n\}$. We have

$$\begin{aligned} \text{Prob}_i(C(i) \in S_i) &= \text{Prob}_i(C(i) \in S_i \mid S_i \text{ is } C\text{-bad}) \cdot \text{Prob}_i(S_i \text{ is } C\text{-bad}) \\ &\quad + \text{Prob}_i(C(i) \in S_i \mid S_i \text{ is not } C\text{-bad}) \cdot \text{Prob}_i(S_i \text{ is not } C\text{-bad}) \\ &\leq 1 \cdot (p_C + \delta_n) + 0 < \frac{2^{-n}}{1 - 2^{-2^{n/4}}} + \frac{2^{-n/4}}{\sqrt{2}} < 2^{-n/4}. \end{aligned}$$

Therefore, we have shown for every circuit C of size $2^{n/4}$ that $\mathrm{Prob}_i(C(i) \in S_i) < 2^{-n/4}$, thus proving the required properties of the sets $S_1, \ldots, S_{2^n}$.

Such a collection can be generated by a Turing machine by considering all possible collections $\{S_1, \ldots, S_{2^n}\}$ and checking whether they evade all the circuits in the set $C^{(n)}$. □

Acknowledgments. Silvio Micali was one of the only researchers to strongly object to the "intuition" that the parallel versions of the interactive proof systems for quadratic residuosity, graph isomophism, and all languages in NP, are zero-knowledge. The results in this paper are rooted in this clear vision of Silvio. We are grateful to him for that.

REFERENCES

[Bab] L. Babai, *Trading group theory for randomness*, in Proc. 17th ACM STOC, 1985, pp. 421–429.

[BMO1] M. Bellare, S. Micali, and R. Ostrovsky, *Perfect zero knowledge in constant rounds*, in Proc. 22nd ACM STOC, 1990, pp. 482–493.

[BMO2] ———, *The (true) complexity of statistical zero knowledge*, in Proc. 22nd ACM STOC, 1990, pp. 494–502.

[B*] M. Ben-Or, O. Goldreich, S. Goldwasser, J. Hastad, J. Killian, S. Micali, and P. Rogaway, *Every thing provable is provable in ZK*, in Advances in Cryptology—Crypto '88 Proceedings, S. Goldwasser, ed., Lecture Notes in Comput. Sci. 403, Springer-Verlag, Berlin, 1989, pp. 37–56.

[BCC] G. Brassard, D. Chaum, and C. Crépau, *Minimum disclosure proofs of knowledge*, J. Comput. Systems Sci., 37 (1988), pp. 156–189.

[BCY] G. Brassard, C. Crépau, and M. Yung, *Everything in NP can be argued in perfect zero-knowledge in a bounded number of rounds*, in Proc. 16th ICALP, Stresa, Italy, 1989.

[CG] B. Chor and O. Goldreich, *On the power of two-point based sampling*, J. Complexity, 5 (1989), pp. 96–106.

[Fei] U. Feige, *Interactive proofs*, M.Sc. thesis, Weizmann Institute, 1987.

[FS1] U. Feige and A. Shamir, *Zero knowledge proofs of knowledge in two rounds*, in Advance in Cryptology—Crypto '89 Proceedings, Lecture Notes in Comput. Sci. 435, G. Brassard, ed., 1989, pp. 526–544.

[FS2] ———, *Witness indistinguishability and witness hiding protocols*, in Proc. 22nd ACM STOC, 1990, pp. 416–426.

[Gol] O. Goldreich, *A uniform-complexity treatment of encryption and zero-knowledge*, J. Cryptology, 6 (1993), pp. 21–53.

[GKa] O. Goldreich and A. Kahan, *How to construct constant-round zero-knowledge proof systems for NP*, J. Cryptology, to appear.

[GK] O. Goldreich and H. Krawczyk, *Sparse pseudorandom distributions*, Random Structures and Algorithms, 3 (1992), pp. 163–174.

[GMW1] O. Goldreich, S. Micali, and A. Wigderson, *Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proofs*, J. Assoc. Comput. Mach., 38 (1991), pp. 691–729.

[GMW2] ———, *How to play any mental game or a completeness theorem for protocols with honest majority*, in Proc. 19th ACM STOC, 1987, pp. 218–229.

[GO] O. Goldreich and Y. Oren, *Definitions and properties of zero-knowledge proof systems*, J. Cryptology, 6 (1993), pp. 1–32.

[GM] S. Goldwasser and S. Micali, *Probabilistic encryption*, J. Comput. System Sci., 28, (1984), pp. 270–299.

[GMR1] S. Goldwasser, S. Micali, and C. Rackoff, *Knowledge complexity of interactive proofs*, in Proc. 17th ACM STOC, 1985, pp. 291–304.

[GMR2] ———, *The knowledge complexity of interactive proof systems*, SIAM J. Comput., 18 (1989), pp. 186–208.

[GS] S. Goldwasser and M. Sipser, *Private coins versus public coins in interactive proof systems*, in Advances in Computing Research: A Research Annual, Vol. 5 (Randomness and Computation, S. Micali, ed.), 1989, pp. 73–90.

[Hoe] W. Hoeffding, *Probability inequalities for sums of bounded random variables*, J. Amer. Statist. Assoc., 58 (1963), pp. 13–30.

[IY] R. Impagliazzo and M. Yung, *Direct minimum-knowledge computations*, in Advances in Cryptology—Crypto '87 Proceedings, C. Pomerance, ed., Lecture Notes in Comput. Sci. 293, Springer-Verlag, 1987, pp. 40–51.

[Jof] A. Joffe, *On a set of almost deterministic k-independent random variables*, Ann. Probab., 2 (1974), pp. 161–162.

[Ore] Y. OREN, *On the cunning power of cheating verifiers: Some observations about zero-knowledge proofs*, in Proc. 28th IEEE Symp. on OCS, 1987, pp. 462–471.

[Sim] D. SIMON, *Issues in the definition of zero-knowledge*, M.Sc. Thesis, University of Toronto, 1988.

[Sha] A. SHAMIR, IP = PSPACE, in Proc. 31st IEEE Symp. on FOCS, 1990, pp. 11–15.

[TW] M. TOMPA AND H. WOLL, *Random self-reducibility and zero-knowledge interactive proofs of possession of information*, in Proc. 28th IEEE Symp. on FOCS, 1987, pp. 472–482.

[WC] M. N. WEGMAN AND J. L. CARTER, *New hash functions and their use in authentication and set equality*, J. Comput. Systems Sci., 22 (1981), pp. 265–279.

[Yao] A. C. YAO, *How to generate and exchange secrets*, in Proc. 27th IEEE Symp. on FOCS, 1986, pp. 162–167.