Proofs that Yield Nothing But Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems

ODED GOLDREICH

Technion, Haifa, Israel

SILVIO MICALI

Massachusetts Institute of Technology, Cambridge, Massachusetts

AND

AVI WIGDERSON

Hebrew University, Jerusalem, Israel

Abstract. In this paper the generality and wide applicability of *zero-knowledge proofs*, a notion introduced by Goldwasser, Micali, and Rackoff is demonstrated. These are probabilistic and interactive proofs that, for the members of a language, efficiently demonstrate membership in the language without conveying any additional knowledge. All previously known zero-knowledge proofs were only for number-theoretic languages in NP ∩ CoNP.

Under the assumption that secure encryption functions exist or by using "physical means for hiding information," it is shown that all languages in NP have zero-knowledge proofs. Loosely speaking, it is possible to demonstrate that a CNF formula is satisfiable without revealing any other property of the formula, in particular, without yielding neither a satisfying assignment nor properties such as whether there is a satisfying assignment in which $x_1 = x_3$ etc.

It is also demonstrated that zero-knowledge proofs exist "outside the domain of cryptography and number theory." Using no assumptions, it is shown that both graph isomorphism and graph nonisomorphism have zero-knowledge interactive proofs. The mere existence of an interactive proof for graph nonisomorphism is interesting, since graph nonisomorphism is not known to be in NP and hence no efficient proofs were known before for demonstrating that two graphs are not isomorphic.

Categories and Subject Descriptors: C.2.0 **[Computer-Communication Networks]**: General—data

---

This work was done while O. Goldreich was at the Laboratory for Computer Science of MIT, and A. Wigderson was at the Mathematical Sciences Research Institute of UC-Berkeley.

This work was partially supported by an IBM Postdoctoral Fellowship, National Science Foundation grants DCR 85-09905 and DCR 84-13577, and an IBM Faculty Development Award.

A preliminary version of this paper appeared in *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1986, pp. 174–187 [42].

Authors' addresses: O. Goldreich, Department of Computer Science, Technion, Haifa, Israel; S. Micali, Laboratory for Computer Science, MIT, Cambridge, MA 02139; A. Wigderson, Institute of Mathematics and Computer Science, Hebrew University, Jerusalem, Israel.

communication, security and protection; C.2.2 [**Computer-Communication Networks**]. Network Protocols; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; D.4.6 [**Operating Systems**]: Security and Protection—*authentication, cryptographic controls, verification*; E.3 [**Data Encryption**]; F.1.2 [**Computation by Abstract Devices**]: Modes of Computation—*alternation and nondeterminism*; *probabilistic computation*; F.1.3 [**Computation by Abstract Devices**]: Complexity Classes—*reducibility and completeness, relation among complexity classes*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*complexity of proof procedures*; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic—*proof theory*; F.4.3 [**Mathematical Logic and Formal Languages**]: Formal Languages—*classes defined by resource-bounded automata (NP)*, G.2.2 [**Discrete Mathematics**]: Graph Theory—*graph algorithms*; G 3 [**Probability and Statistics**]: *probabilistic algorithms*; G.4 [**Mathematical Software**]: *reliability and robustness, verification*; K.4.1 [**Computers and Society**]: Public Policy Issues—*privacy*; K.6.m [**Management of Computing and Information Systems**]—*security*

General Terms: Design, Security, Theory, Verification

Additional Key Words and Phrases: Cryptographic protocols, fault-tolerant distributed computing, graph isomorphism, interactive proofs, methodological design of protocols, NP, one-way functions, proof systems, zero-knowledge

1. *Introduction*

It is traditional to view NP as the class of languages whose elements possess short proofs of membership. A "proof that $x \in L$" is a witness w_x such that $P_L(x, w_x) = 1$ where P_L is a polynomial-time computable Boolean predicate associated to the language L(i.e., $P_L(x, y) = 0$ for all y if $x \notin L$). The witness must have length polynomial in the length of the input x, but need not be computable from x in polynomial-time. A slightly different point of view is to consider NP as the class of languages L for which a powerful prover may prove membership in L to a deterministic polynomial-time verifier. The interaction between the prover and the verifier, in this case, is trivial: the prover sends a witness (proof) and the verifier computes for polynomial-time to verify that it is indeed a proof.

This formalism was recently generalized by allowing more complex interaction between the prover and the verifier and by allowing the verifier to toss coins and to be convinced by overwhelming statistical evidence [6, 47]. The prover has some advantage over the verifier and, for the definition to be interesting, one should assume that this advantage is crucial for proving membership in the language (otherwise, the verifier can do this by itself). This advantage may take either the form of much stronger computing power or extra information concerning the input. In other words, we implicitly assume that there exist interesting languages (say in PSPACE) that are not in BPP, and be interested in proof systems for such languages.

A fundamental measure proposed by Goldwasser et al. [47] is that of the amount of knowledge released during an interactive proof. Informally, a proof system is called zero-knowledge if whatever the verifier could generate in probabilistic polynomial-time after "seeing" a proof of membership, he could also generate in probabilistic polynomial-time when just told by a trusted oracle that the input is indeed in the language. Zero-knowledge proofs have the remarkable property of being both convincing and yielding nothing except that the assertion is indeed valid.

Besides being a very intriguing notion, zero-knowledge proofs promise to be a very powerful tool for the design of secure cryptographic protocol. Typically these protocols must cope with the problem of distrustful parties convincing

each other that the messages they are sending are indeed computed according to their predetermined local program. Such proofs should be carried out without yielding any secret knowledge. In particular cases, zero-knowledge proofs were used to design secure protocols [23, 30, 47]. However, in order to demonstrate the generality of this tool (and to utilize its full potential) one must present general results concerning the existence of zero-knowledge proof systems. Until now, no such general results were known.

In this paper, we present general results concerning zero-knowledge proof systems. In particular, we show how to construct zero-knowledge proofs for every NP-statement. This result has a dramatic effect on the design of cryptographic protocols: it constitutes the keystone in a methodology for cryptographic protocol design. The methodology, presented in another paper of the authors [43], consists of a "privacy and correctness preserving" compiler, which translates protocols for a weak adversarial model to a fully fault-tolerant protocol (withstanding the most adversarial behavior).

1.1 What Is an Interactive Proof? Intuitively, an interactive proof system for a language L is a two-party protocol for a "powerful" *prover* and a probabilistic polynomial-time *verifier* satisfying the following two conditions with respect to the common input, denoted x. If $x \in L$, then, with very high probability, the verifier is "convinced" of this fact, after interacting with the prover. If $x \notin L$, then no matter what the prover does, with very high probability, he fails to fool the verifier (into believing that "x is in L"). The first condition is referred to as the *completeness* condition, while the second condition is referred to as *soundness*.

Before defining the notion of an interactive proof system, we define what we mean by "expected polynomial-time," and recall the notion of an interactive pair of Turing machines, which captures the intuitive notion of a two-party protocol.

Definition and Discussion of Expected Polynomial-Time. Let M be a probabilistic Turing machine, and $t_M(x, r)$ denote the number of steps taken by machine M on input x when the outcome of its internal coin tosses is a prefix of the infinite bit sequence r. Clearly, if $t_M(x, r)$ is finite it depends only on a finite prefix of r. Traditionally, machine M is called "expected polynomial-time" if there exists a polynomial Q such that, for all $x \in \{0, 1\}^*$, the expectation, taken over the infinite bit sequences r, of $t_M(x, r)$ is bounded above by $Q(|x|)$ (i.e., $E_r(t_M(x, r)) \leq Q(|x|)$, for all x). One can use this definition in our paper, however we would like to call the reader's attention to weaknesses of this definition that were pointed out by Levin (see also [38] and [50]).[1] We prefer to follow Levin's definition [57], saying that a machine M is *expected polynomial-time* if there exists an $\epsilon > 0$ such that, for all $x \in \{0, 1\}^*$, the expectation, taken over the infinite bit sequences r, of $(t_M(x, r))^\epsilon$ is bounded above by $|x|$ (i.e., $E_r((t_M(x, r))^\epsilon) \leq |x|$). We say that M is (strictly) *probabilistic polynomial-time* if $t_M(x, r) < Q(|x|)$ for some polynomial Q and all x, r pairs.

[1] For example, $E_r(t(x, r)) \leq Q(|x|)$ does not imply $E_r(t^2(x, r)) \leq Q(|x|)^2$, and problems as machine dependency and nonclosure under reduction, of expected polynomial-time machines, follow.

Definition 1. (Interactive Turing machines (M. Blum), private communication):

(1) An *interactive Turing machine* (*ITM*) is a six-tape deterministic Turing machine with a read-only *input tape*, a read-only *random tape*, a read/write *work tape*, a read-only *communication tape*, a write-only *communication tape*, and a write-only *output tape*. The string that appears on the input tape is called the *input*. The (infinite) contents of the random tape can be thought of as the outcomes of an infinite sequence of unbiased coin tosses. The string that appears on the output tape when the machine halts is called the *output*. The contents of the write-only communication tape can be thought of as *messages sent* by the machine; while the contents of the read-only communication tape can be thought of as *messages received* by the machine.

(2) The *complexity of an interactive Turing machine* is measured in terms of its input (i.e., contents of input tape). Saying, for example, that the interactive Turing machine M is polynomial-time means that there exists a polynomial Q such that the number of steps M performs on input x is at most $Q(|x|)$, no matter what are the contents of its random tape and read-only communication tape. When discussing the expected number of steps of an ITM, the expectation is taken over the contents of the random tape only. (We do not count, of course, the steps made by the interactive machine with which M interacts as defined below.)

(3) An *interactive pair of Turing machines* is a pair of ITMs that share their communication tapes so that the read-only communication tape of the first machine coincides with the write-only communication tape of the second machine, and vice versa. The computation of such a pair consists of alternating sequences of computing steps taken by each machine. The alternation occurs when the *active* machine enters a special *idle* state. At this time, the other machine passes from *idle* to the *active* state. The string written on the communication tape during a single nonalternating sequence of steps is called the *message sent* by the active ITM to the idle one.

Notations

(1) Let A be an ITM. Then $A(x, r; \alpha_1, \alpha_2, \ldots, \alpha_t)$ denotes the message sent by A on input x, random tape contents r, after receiving the messages α_1 through α_t.
(2) Let A and B be an interacting pair of ITMs. Then, $[B(y), A(x)]$ denotes the output distribution of A on input x, when B has input y. The probability space is induced by the unbiased coin tosses of both machines.

Definition 2 (Interactive Proof System [47]). An *interactive proof system for a language* L is a pair of ITMs, $\langle P, V\rangle$, such that V is expected polynomial-time and the following two conditions hold:

(1) *Completeness Condition*. For every constant $c > 0$, and all sufficiently long $x \in L$

$$\Pr\left(\left[P(x), V(x)\right] = 1\right) \geq 1 - |x|^{-c}.$$

(2) *Soundness Condition*. For every constant $c > 0$, every interactive Turing machine P^*, and all sufficiently long $x \notin L$,

$$\Pr\left(\left[P^*(x), V(x)\right] = 0\right) \geq 1 - |x|^{-c}.$$

Denote by IP the class of languages having interactive proof systems.

Remark 1. Clearly, NP $\subseteq$ IP. An example of an interactive proof system, for a language not known to be in NP, is presented in Section 2.1.

Remark 2. As is the case with NP, the conditions imposed on acceptance and rejection are not symmetric. Thus, the existence of an interactive proof system for the language L *does not* imply the existence of an interactive proof system for the complement of L.

Remark 3. Definition 2 originates from Goldwasser et al. [47]. In [6], Babai introduced a different type of interactive proof systems, called *Arthur–Merlin games*, in which the verifier's behavior is more restricted than in the Goldwasser et al. model [47]. Arthur, the verifier in Babai's model, is allowed only to generate random strings, send them to the prover, and evaluate a deterministic polynomial-time predicate at the end of the interaction. Demonstrating the existence of proof systems is easier when allowing the verifier to flip private coins (i.e., Goldwasser et al. model [47]), while relating interactive proof systems to traditional complexity classes seems easier if one restricts oneself to Arthur–Merlin games. Interestingly, as Goldwasser and Sipser showed, these two models are equivalent, as far as language recognition is concerned [49]. Namely, let L be a language having an interactive proof system in which up to $Q(|x|)$ messages are exchanged on input $x \in L$. Then, L has an Arthur–Merlin interactive proof system in which up to $Q(|x|) + 2$ messages are exchanged on input $x \in L$.

Remark 4. The ability to toss coins is crucial to the nontriviality of the notion of an interactive proof system. If the verifier is deterministic, then interactive proof systems coincide with NP.

Remark 5. The fact that the verifier is expected polynomial-time instead of being strictly probabilistic polynomial-time has no effect on the notion of interactive proofs. However, as seen in Remarks 11 and 13, allowing the simulator (see next subsection) to be expected polynomial-time plays an important role in the results concerning *perfect* zero-knowledge proof systems.

Babai and Moran [8] showed that, for every polynomial Q, AM($2Q$) = AM(Q), where AM(Q) denotes the class of languages recognized by an Arthur–Merlin game of $Q(|\cdot|)$ message exchanges (last message sent by M). This means that the finite level Arthur–Merlin hierarchy (as well as the finite level IP hierarchy) collapses (i.e., for every fixed $k \geq 2$, AM(k) = AM(2); hereafter abbreviated AM). Note that this does not imply the collapse of the unbounded level hierarchy! (For more details, see [2].) The bounded level interactive proofs hierarchy is related to the polynomial-time hierarchy by Babai's proof that AM $\subseteq \Pi_2^P$ (the second level of the polynomial-time hierarchy) and that AM $\subseteq$ NP^B for almost all oracles B. Recently, Nissan and Wigderson showed that AM = NP^B for almost all oracles B [59].

1.2 What Is a Zero-Knowledge Proof? Intuitively, a zero-knowledge proof is a proof that yields nothing but its validity. This means that for all practical purposes, "whatever" can be efficiently computed after interacting with a zero-knowledge prover and can be efficiently computed when just believing that the assertion it claims is indeed valid. (In "whatever" we mean not only the computation of functions but also the generation of probability distributions.) Zero-knowledge is a property of the predetermined prover: Its

robustness against attempts of the verifier to extract knowledge via interaction. Note that the verifier may deviate arbitrarily (but in polynomial-time) from the predetermined program. This is captured by the following formulation:

Definition 3 (Zero-Knowledge [47]). Let $\langle P, V\rangle$ be an interactive proof system for a language L. We say that the proof system $\langle P, V\rangle$ is *zero-knowledge* (for L) if for every expected polynomial-time interactive Turing machine V^*, there exists an (ordinary) expected polynomial-time machine M_{V^*} such that the probability ensembles $\{M_{V^*}(x)\}_{x\in L}$ and $\{[P(x), V^*(x)]\}_{x\in L}$ are polynomially indistinguishable (see Definition 4 below).

The machine M_{V^*} is called the *simulator* of V^*. The simulator of V^* has essentially the same output distribution as V^*; however, M_{V^*} produces this distribution without interacting with the prover (while V^* produces it with the help of interaction with the prover). The universal quantifier states that this can be done with respect to every efficient way (not necessarily the proper one) of interacting with the prover. It is not required that the simulator can be effectively constructed given a verifier, but rather that it exists. Nevertheless, all known demonstrations (including ours) of the zero-knowledge property of protocols are constructive.

The simulator, M_{V^*}, needs not generate exactly the ensemble that V^* outputs when interacting with the prover. It suffices to produce something that is indistinguishable to polynomial-time computations. (This is the reason that zero-knowledge, as defined above, is sometimes called computational zero-knowledge, especially in order to distinguish it from stronger forms of zero-knowledge–see Remark 7.) For the sake of self-containment, we recall the definition of polynomial indistinguishable ensembles.

Definition 4 (Polynomial Indistinguishability [46, 68]). Let $S \subseteq \{0,1\}^*$ be an infinite set of strings, and let $\Pi_1 = \{\Pi_1(x)\}_{x\in S}$ and $\Pi_2 = \{\Pi_2(x)\}_{x\in S}$ be two probability ensembles (i.e., for every $i\in\{1,2\}$ and $x\in S$, $\Pi_i(x)$ is a random variable assuming values in $\{0,1\}^*$). For every algorithm (test) A, let $p_i^A(x)$ denote the probability that A outputs 1 on input x and an element chosen according to the probability distribution $\Pi_i(x)$. Namely,

$$p_i^A(x) = \sum_{\alpha} \Pr\big(A(x,\alpha) = 1\big)\cdot \Pr\big(\Pi_i(x) = \alpha\big).$$

The ensembles $\Pi_1 = \{\Pi_1(x)\}_{x\in S}$ and $\Pi_2 = \{\Pi_2(x)\}_{x\in S}$ are *polynomially indistinguishable* if for every expected polynomial-time algorithm A, for every constant $c > 0$ and for all sufficiently long $x\in S$

$$|\, p_1^A(x) - p_2^A(x)\,| \le |\, x\,|^{-c}.$$

Polynomially indistinguishable probability ensembles should be considered equal for all practical purposes, since any application running in polynomial-time and using either ensemble demonstrates essentially the same behavior. It follows that the polynomial indistinguishability of $[P(x), V^*(x)]$ and $M_{V^*}(x)$ formalizes the intuition that nothing *substantial* is gained by interacting with the prover, except of course conviction in the validity of the assertion $x\in L$.

An alternative definition of zero-knowledge considers the probability distribution on all the tapes of V^* during the interaction with P. In fact, it suffices to consider the contents of all the read-only tapes of V^* (i.e., the input tape, the random tape, and the read-only communication tape of V^*).

Definition 5 (Zero-Knowledge–alternative definition). Let $\langle P, V \rangle$ be an interactive proof system for a language L, and V^* be an arbitrary ITM. Denote by $\langle P, V^* \rangle(x)$ the probability distribution on all the read-only tapes of V^*, when interacting with P (the prover) on common input $x \in L$. We say that the proof system $\langle P, V \rangle$ is *zero-knowledge* (for L) if for all expected polynomial-time ITM V^*, there exists an expected polynomial-time machine M_{V^*} such that the probability ensembles $\{M_{V^*}(x)\}_{x \in L}$ and $\{\langle P, V^* \rangle(x)\}_{x \in L}$ are polynomially indistinguishable.

Remark 6. It is easy to see that Definitions 3 and 5 are in fact equivalent. The output of V^* is easily computed from the contents of its read-only tapes (i.e., $\forall V^* \; \exists T^*$ such that $[P(x), V^*(x)] = T^*(\langle P, V^* \rangle(x))$). On the other hand, every V^* can be *modified* easily to output the contents of its read-only tapes (i.e., $\forall V^* \; \exists V^{**}$ such that $\langle P, V^* \rangle(x) = [P(x), V^{**}(x)]$). Hence, ability to simulate the output of every V^* implies ability to simulate the contents of the tapes of every V^*. However, it is not always the case that ability to simulate the output of a *specific* V^* implies ability to simulate the contents of V^*'s tapes. This difference is best demonstrated by considering the prescribed verifier V. Consider for example a proof system, $\langle P, V \rangle$, for an NP-complete language L, in which P sends to V a witness for membership which V tests using the witness predicate P_L. Then, it is trivial to simulate $\{[P(x), V(x)]\}_{x \in L}$ which is identically 1, while it is probably impossible to simulate $\langle P, V \rangle(x)$ (unless NP $\subseteq$ BPP).

Remark 7. In case one can exactly simulate the probability distributions of the interaction with the prover, rather than produce distributions that are polynomially indistinguishable from them, we say that the interactive proof system is *perfect* zero-knowledge. Formally, an interactive proof system for L is *perfect zero-knowledge* if, for every expected polynomial-time ITM V^*, there exists an expected polynomial-time machine M_{V^*} such that for every $x \in L$

$$\langle P, V^* \rangle(x) = M_{V^*}(x).$$

It is worth noting that in the definition of "perfect zero-knowledge" the use of expected polynomial-time seems to be more crucial than for the other types of zero-knowledge (i.e., computational and almost-perfect): see Remarks 11, 13, and 16. An interactive proof system is *almost-perfect zero-knowledge* if for every $x \in L$ the distributions $\langle P, V^* \rangle(x)$ and $M_{V^*}(x)$ are "statistically close", that is, the statistical difference (defined below) between $\langle P, V^* \rangle(x)$ and $M_{V^*}(x)$ is smaller than $|x|^{-c}$, for all $c > 0$ and sufficiently large x. The *statistical difference* (also called variation distance) between two probability distributions is the sum of the absolute differences in the probability mass assigned by these distributions to each of the possible elements. An equivalent definition can be derived from the definition of polynomial-indistinguishable distributions, when omitting the restriction on the running-time of the algorithm A (required above to be expected polynomial-time).

Remark 8. Our model of interactive proof systems is implicitly asynchronous (i.e., an interactive pair of Turing machines formalizes the standard notion of an *event-driven* protocol for two parties). Thus, one machine cannot infer the number of steps taken by its counterpart from the "delay" between the

communications. In real applications, such inference may be possible, and an easy modification of the definition of zero-knowledge will be adequate.

Remark 9. It is not difficult to see that if a language L has a zero-knowledge proof system in which a single message is sent then L is in Random polynomial-time [44, 60]. Thus, the generalization of NP to IP is essential to the nontriviality of the notion of zero-knowledge.

Clearly, every language that is BPP has a (perfect) zero-knowledge interactive proof system (in which the prover does nothing). Several number-theoretic languages, *not known to be in* BPP, have been shown to have zero-knowledge proof systems. The first language for which such a proof system was demonstrated is Quadratic Non-Residuosity [47]. Other zero-knowledge proof systems were presented in [23], [32], [36], and [47]. All these languages are known to lie in NP $\cap$ CoNP.

1.3 Our Results. Using any scheme for "committing to a secret bit", we present zero-knowledge proof systems for all languages in NP. A scheme for "committing to a secret bit" can be implemented either by using any one-way function (assumed to exist) or by using "physical means for hiding information." This result demonstrates the generality of zero-knowledge proofs, and is the key for their wide applicability to cryptography and related fields.

We also demonstrate that zero-knowledge proof systems exist "independently of cryptography and number theory." Using no assumptions, we show that both graph isomorphism and graph nonisomorphism have perfect zero-knowledge interactive proof systems.

1.4 Related Results. Using the intractability assumption of *quadratic residuosity*, Brassard and Crépeau have subsequently discovered zero-knowledge proof systems for all languages in NP [16]. These proof systems heavily rely on *particular properties of quadratic residues* and do not seem to extend to arbitrary bit commitment schemes.

Independently of our work, Brassard et al. showed that, *if factoring is intractable*, then every NP language has a perfect zero-knowledge *argument*[2] [15, 17, 19]. The difference between an *argument* and interactive proof systems is that in an argument the soundness condition is restricted to probabilistic polynomial-time machines (with auxiliary input which is fixed before the protocol starts).[3] Hence it is *infeasible* (not impossible) to fool the verifier into accepting (with nonnegligible probability) an input not in the language. Brassard et al. also proposed an interesting application of perfect zero-knowledge arguments to settings in which the verifier may have infinite computing power while the prover is restricted to polynomial-time computations. In such a setting, it makes no sense to have the prover demonstrate

[2] The term "argument" first appeared in [18]. In the original papers of Brassard et al. [15, 17, 19], the authors referred to *arguments* by the term interactive proofs, thus creating an enormous amount of confusion. These two notions are probably distinct (otherwise, some drastic consequences to complexity theory occur). For example, the results of Fortnow [31] and Aiello and Hastad [3], cited below, do not hold for arguments. In some works arguments are called "computationally-sound proofs".

[3] The reader should not confuse arguments (in which soundness depends on a computational restriction imposed on "cheating provers") and interactive proof systems with the additional property that the prescribed prover may be a probabilistic polynomial-time machine with an appropriate auxiliary input (e.g., Protocols 2 and 4).

properties (as membership in a language) to the verifier. However, the prover may wish to demonstrate to the verifier that he "knows" something without revealing what he "knows." More specifically, given a CNF formula, the prover may wish to convince the verifier that he "knows" a satisfying assignment in a manner that would yield no information as to which of the satisfying assignments he knows. A definition of the notion of "a program knowing a satisfying assignment" was first sketched in [47], and recently formalized in [27] and [67].

1.5 Some Subsequent Results Concerning Interactive Proof Systems and Zero-Knowledge. Subsequent results place languages with perfect zero-knowledge proof systems quite low in the polynomial-time hierarchy. More specifically, if a language L has a perfect (or even almost-perfect) zero-knowledge proof system then $L \in \mathrm{AM} \cap \mathrm{CoAM}$ (the $L \in \mathrm{CoAM}$ implication was shown by Fortnow [31] and subsequently Aiello and Hastad showed the $L \in \mathrm{AM}$ part [3]). It follows that, unless all CoNP languages have interactive proof systems, NP-Complete languages cannot have perfect (or almost-perfect) zero-knowledge proof systems. It is interesting to note that Fortnow gives a transformation of a perfect zero-knowledge proof system for L into an interactive proof system for $\overline{L}$, that when applied to our perfect zero-knowledge protocol for graph isomorphism yields exactly our interactive proof system for Graph Nonisomorphism.

Recently, it was shown that the error probability allowed in the completeness condition of interactive proofs is not essential for the power of interactive proof systems [41]. On the other hand, the error probability allowed in the soundness condition is essential to the nontriviality of both interactive proof systems and zero-knowledge. Interactive proof systems with no error on NO-instances equal NP [41], while zero-knowledge proof systems with no error on NO-instances equal R [44, 60]. It follows that the error probability on the NO-instances of the protocols presented in this paper (e.g., the zero-knowledge protocols for NP and the interactive proof of Graph nonisomorphism) cannot be avoided, unless something dramatic happens (e.g., R = NP or graph nonisomorphism is in NP, respectively).

Our result that all languages in NP have zero-knowledge proof systems, has been extended to IP, assuming the same assumptions. (The result was first proved by Impagliazzo and Yung, but since their paper [53] contains only a claim of the result, the interested reader is directed to [11] where a (different) proof appears.) In other words, whatever can be efficiently proven can be efficiently proven in a zero-knowledge manner. This *may be viewed* as the best result possible, since only languages having interactive proof systems can have zero-knowledge interactive proof systems.

1.6 Conventions. Let A be a finite set. Then $Sym(A)$ denotes the symmetric group of A (i.e., the group of permutations over the set A). Let $\pi, \phi \in Sym(A)$, then $\pi \cdot \phi$ denotes the composition of the permutations π and ϕ (i.e., $\pi \cdot \phi(x) = \pi(\phi(x))$).

When writing $a \in_R A$, we mean an element chosen at random with uniform probability distribution from the set A. If $|A|$ is a power of 2, then such a random selection is easily implemented in the standard probabilistic model (by tossing $\log_2 |A|$ unbiased coins). Otherwise, a random choice can be implemented in the standard model using an expected number of $O(\log |A|)$

(unbiased) coin tosses. If one wants to avoid infinite (or long) runs, such a random choice can only be approximated (very well) and all the constructions we use have to be slightly modified. When writing $\Pr_{a \in A}(P(a))$, we mean the probability that $P(a)$ holds when the probability is taken over all choices of $a \in A$ with uniform probability distribution.

The notation $|\cdot|$ is used in three different ways. As denoting the cardinality of a set, the length of a string, and the absolute value of a real. We trust that the reader will figure out the meaning by the context.

We consider simple (i.e., no parallel edges) undirected graphs. By V, we denote the vertex set, and by E the edge set of the graph G. Let n denote the cardinality of the vertex set, and m the cardinality of the edge set (i.e., $n = |V|$, $m = |E|$). The graph $G(V, E)$ will be represented by the set E, in an arbitrary fixed order (e.g., lexicographic).

In all our protocols the error probability (in the soundness condition) is exponentially decreasing with m and thus negligible in the input size. In fact, it would have sufficed to make the error probability decrease faster than $1/m^c$ for $c > 0$.

1.7 Organization of the Paper. In Section 2, we present zero-knowledge interactive proof systems for graph isomorphism and graph nonisomorphism. These protocols serve as good examples of the notions of interactive proof systems and zero-knowledge proof systems, and of the techniques used to prove that a protocol indeed satisfies these properties. We also discuss complexity theoretic implications of the existence of an interactive proof system for graph nonisomorphism.

In Section 3, we show how to use any bit commitment scheme in order to construct a zero-knowledge interactive proof system for any language in NP.

2. *Proofs Systems for Graph Isomorphism and Graph Nonisomorphism*

We start by presenting a (probably nonzero-knowledge) interactive proof system for graph nonisomorphism. Next we present a zero-knowledge interactive proof system for graph isomorphism, and for graph nonisomorphism.

Two graphs $G(V, E)$ and $H(V, F)$ are *isomorphic* if and only if there exists a permutation $\pi \in Sym(V)$ such that $(u, v) \in E$ iff $(\pi(u), \pi(v)) \in F$. We then write $H = \pi G$. We say that the graph $H(V, F)$ is a *random isomorphic copy* of the graph $G(V, E)$ if H is obtained from G by picking $\pi \in_R Sym(V)$ and letting $H = \pi G$.

The language GI (*graph isomorphism*) consists of all the pairs of isomorphic graphs (i.e., $GI = \{(G, H): \exists \pi \text{ such that } H = \pi G\}$). Its complement, *graph nonisomorphism* (GNI), is the set of all nonisomorphic pairs.

The language GI is in NP, is not known to be in CoNP, and is believed not to be NP-complete. The fastest algorithm known for the corresponding (search and decision) problems runs in time exponential in $\sqrt{n \log n}$ [7].

2.1 An Interactive Proof System of Graph Nonisomorphism. In this section, we exemplify the notion of an interactive proof system by presenting an interactive proof system for graph nonisomorphism. The fact that graph nonisomorphism has an interactive proof system is interesting as it is not known to be in NP, and hence is not known to have efficient "noninteractive" proof systems. Moreover, the existence of an interactive proof system for graph nonisomorphism has interesting complexity-theoretic consequences.

In the following protocol, the prover may be a probabilistic polynomial-time machine with access to an oracle for graph isomorphism.

Protocol 1.

common input. Two graphs $G_1(V, E_1)$ and $G_0(V, E_0)$.

(1) The verifier chooses at random m bits, $\alpha_i \in_R \{0, 1\}$, $1 \le i \le m$. The verifier computes m graphs H_i such that H_i is a random isomorphic copy of G_{α_i}. The verifier sends the H_i's to the prover.
(2) The prover answers with a string of β_i's (each in $\{0, 1\}$), such that H_i is isomorphic to G_{β_i}.
(3) The verifier tests whether $\alpha_i = \beta_i$, for all $1 \le i \le m$. If the condition is violated then the verifier *rejects*; otherwise it *accepts*.

THEOREM 1. *Protocol* 1 *constitutes a (two-move) interactive proof system for Graph Nonisomorphism.*

PROOF. If the graphs G_1 and G_0 are not isomorphic, and both prover and verifier follow the protocol, then the verifier always accepts. If on the other hand, G_1 and G_0 are isomorphic then, for each i, we have $\alpha_i \neq \beta_i$ with probability at least $\frac{1}{2}$, even if the prover does not follow the protocol.[4] This is because, if G_1 and G_0 are isomorphic, then

$$\Pr(\alpha_i = 1 \mid \text{verifier sent } H_i) = \frac{1}{2}.$$

The probability that the verifier does not reject two isomorphic graphs is thus at most 2^{-m}. □

The above theorem has interesting consequences on the computational complexity of the graph isomorphism problem. Namely,

COROLLARY 1. *Graph Isomorphism is in* $(NP \cap CoNP)^A$, *for a random oracle* A. *Also, Graph Nonisomorphism can be recognized by a (nonuniform) family of nondeterministic polynomial-size circuits (i.e.,* $GNI \in NP/poly$).

PROOF. By Theorem 1, $GNI \in \mathrm{IP}(2)$. Using Goldwasser and Sipser's transformation of $\mathrm{IP}(k)$ protocols to $\mathrm{AM}(k+2)$ protocols, $GNI \in \mathrm{AM}(4)$. By Babai's proof of the finite $\mathrm{AM}(\cdot)$ collapse, $GNI \in \mathrm{AM} \subseteq \mathrm{NP}^A$ for a random oracle A. Finally, it has been pointed out by Mike Sipser that AM is contained in nonuniform NP (the proof[5] is analogous to the proof that $\mathrm{BPP} \subseteq \mathrm{P/poly}$). □

COROLLARY 2. *If Graph Isomorphism is NP-Complete, then the polynomial-time hierarchy collapses to its second level.*

[4] The probability is exactly $\frac{1}{2}$ if the prover answers with $\beta_i \in \{0, 1\}$, and 0 if (the prover is foolish enough to answer with) $\beta_i \notin \{0, 1\}$.

[5] To prove that $\mathrm{AM} \subseteq \mathrm{NP/poly}$, we consider a two-step Arthur–Merlin proof system for a language $L \in \mathrm{AM}$. Without loss of generality, the error probability on (both "yes" and "no") instances of length n is bounded above by 2^{-n} (this can be achieved by running $O(n)$ copies of the game in parallel and ruling by majority). It follows that there exists a sequence r of coin tosses (for Arthur) such that, for all $x \in \{0, 1\}^n$, $x \in L$ if and only if there exists (an answer of Merlin) $y \in \{0, 1\}^*$ such that Arthur accepts x in the conversation (r, y).

PROOF. Boppana et al. showed that if CoNP $\subseteq$ AM, then the entire polynomial-time hierarchy collapses to AM $\subseteq \Pi_2^P$ [14]. Since Theorem 1 states that $GNI \in$ IP(2), assuming that *GNI* is CoNP-Complete yields CoNP $\subseteq$ IP(2), and the Corollary follows. □

An alternative proof to Corollary 2 was obtained independently by Schoening, using totally different techniques [64]. Corollary 2 may be viewed as providing support for the belief that Graph Isomorphism is not NP-Complete.

2.2 A ZERO-KNOWLEDGE PROOF FOR GRAPH ISOMORPHISM. In this section, we exemplify the notion of zero-knowledge proof systems by presenting a zero-knowledge proof system for graph isomorphism. The mere fact that graph isomorphism has efficient proof systems is clear, as it is in NP. However, the fact that graph isomorphism can be proved in zero-knowledge, and in particular without revealing the isomorphism, is interesting.

In the following protocol, the prover needs only be a probabilistic polynomial-time machine that gets, as an auxiliary input, an isomorphism between the input graphs.

Protocol 2

common input. Two graphs $G_1(V, E_1)$ and $G_0(V, E_0)$.

Let ϕ denote the isomorphism between G_1 and G_0 (i.e., $G_1 = \phi G_0$). The following four steps are executed m times, each time using independent random coin tosses.

(P1) The prover generates a graph H, which is a random isomorphic copy of G_1. This is done by selecting a permutation $\pi \in_R Sym(V)$, and computing $H = \pi G_1$. The prover sends the graph H to the verifier.

(V1) The verifier chooses at random $\alpha \in_R \{0, 1\}$, and sends α to the prover. (Intuitively, the verifier asks the prover to show him that H and G_α are indeed isomorphic.)

(P2) If $\alpha = 1$, then the prover sends π to the verifier, else (i.e., $\alpha \neq 1$) the prover sends $\pi \cdot \phi$. (Note that the case $\alpha \notin \{0, 1\}$ is handled as $\alpha = 0$.[6])

(V2) If the permutation (ψ) received from the prover is not an isomorphism between G_α and H (i.e., $H \neq \psi G_\alpha$), then the verifier stops and *rejects*; otherwise, he continues.

If the verifier has completed m iterations of the above steps, then he *accepts*.

The reader can easily verify that the above constitutes an interactive proof system for graph isomorphism. Intuitively, this proof system is zero-knowledge since whatever the verifier receives is "useless," because he can generate random isomorphic copies of the input graphs by himself. This is easy to see in case the verifier follows the protocol. In case the verifier deviates from the protocol, the situation is much more complex. The verifier may set the α's depending on the graphs presented to him. In such a case, it cannot be argued that the verifier only receives random isomorphic copies of the input graph. The

[6] It may be considered more natural to have the prover halt in case $\alpha \notin \{0, 1\}$ (as done in an earlier version [42]). However, demonstrating that the above version is zero-knowledge is easier since it avoids dealing separately with the case $\alpha \notin \{0, 1\}$ (in the simulation).

issue is fairly involved, as we have to defeat a universal quantifier that is not well understood (i.e., all possible polynomial-time deviations from the protocol). We cannot really trust our intuition in such matters, so a formal proof is indeed required.

THEOREM 2. *Protocol 2 constitutes a (perfect) zero-knowledge interactive proof system for Graph Isomorphism.*

PROOF. We start by proving that Protocol 2 is an interactive proof system for graph isomorphism. First, note that if the input graphs are isomorphic, then the prover can easily supply the permutations required by the verifier. In case the graphs are not isomorphic, then, no matter what the prover does, the graphs H sent by him (in step P1) cannot be isomorphic to both G_1 and G_0. It follows that when asked (in step P2) to present an isomorphism between H and G_α, where $\alpha \in_R \{0, 1\}$, the prover fails with probability $\geq \frac{1}{2}$.

We now come to the difficult part of this proof: demonstrating that Protocol 2 is indeed zero-knowledge. It is easy to see that the prover conveys no knowledge to the *specified* verifier. We need, however, show that our prover conveys no knowledge to all possible verifiers, including cheating ones that deviate arbitrarily from the protocol. In our proof we use Definition 5.

Let V^* be an arbitrary, fixed, expected polynomial-time interactive machine. We will present an expected polynomial-time machine M_{V^*} that generates a probability distribution which is identical to the probability distribution induced on V^*'s tapes during its interaction with the prover. In fact it is more convenient to include in both distributions also the contents of the write-only communication tape of V^*.

Our demonstration of the existence of such an M_{V^*} is constructive: given an interactive machine V^*, we use it in order to construct the machine M_{V^*}. Machine M_{V^*} uses V^* as a subroutine (many times) while placing strings of its choice on all the read-only tapes of V^*, and reading the intermediate and final contents of V^*'s write-only tapes. In particular, M_{V^*} chooses the input to V^*, the contents of the random tape of V^*, and the messages for the read-only communication tape of V^*; and reads the write-only communication tape of V^*. Intuitively speaking, M_{V^*} tries to guess which isomorphism the machine V^* will ask to check. It (i.e., M_{V^*}) constructs the graph H such that it can answer V^* in case it were lucky. This is done by constructing a random isomorphic copy of G_1 starting with either G_1 or with G_0. Such a lucky case becomes part of the output of M_{V^*}. The cases in which M_{V^*} fails are ignored (i.e., do not appear in its output). It is crucial that, from the point of view of V^*, the case which leads to M_{V^*} success and the case which leads to a failure look identical. By throwing away the instances where we failed, we only slow down our construction, but we do not change the probability distribution that V^* "sees."

Following is a detailed description of the simulating machine M_{V^*}. In all invocations of V^*, machine M_{V^*} will place $x = (G_1, G_0)$ on the input tape of V^* and a fixed sequence of randomly chosen bits on its random tape. So, the first thing M_{V^*} does is choose at random the contents of the random tape of the interactive machine V^*, and fix it for the rest of the simulation. Fixing the contents of the random tape of V^* makes its actions depend only on the contents of its read-only communication tape that will be written by M_{V^*}.

There is a subtle problem in implementing this plan, to be discussed in the following paragraph.

The easy case is when V^* is strictly polynomial-time. Namely, all the runs of V^* on input x take at most $Q(|x|)$ steps (independent of the contents of the random tape!), where Q is a fixed polynomial. In this case choosing a random input amounts to choosing $r \in_R \{0,1\}^{Q(|x|)}$. However, in the general case V^* is (only) expected polynomial-time, and there may be no bound on the number of coin tosses it may use on a particular input x. In this case, we need to choose at random an infinite bit sequence. Fortunately, this needs not be done a priori. We select the random input of V^* adaptively. Intuitively, each time V^* requires a new random bit, we select one and store it for all future executions of V^* on x. Formally, M_{V^*} selects $r_i \in_R \{0,1\}$ the first time that V^* tries to scan the ith cell in the random tape in some execution on input x. In all future runs on V^* on x, the contents of the ith cell of the random tape of V^* is initialized to r_i (and never changed!). We denote by r the imaginary infinite sequence of bits chosen (and fixed) for V^* by M_{V^*} in a halting computation on input x. Note that in any finite execution of V^*, only a finite prefix of r is scanned, and only this prefix will be output by M_{V^*} as part of its output. Jumping ahead we stress that M_{V^*} will conduct additional coin tosses, which it will never present to V^*.

The output of machine M_{V^*} will be compiled on special *record tapes*: the *random-record tape* and the *communication-record tape*. The random-record tape will contain the prefix of r scanned by V^* (on input x, random tape contents r, and the conversation appearing on the communication-record tape). The contents of the communication-record tape is constructed in the following m rounds (initially the communication-record is empty and $i = 1$).

Round i:

(S1) M_{V^*} chooses at random $\beta \in_R \{0,1\}$ and a permutation $\psi \in_R Sym(V)$. It computes $H = \psi G_\beta$.

(S2) Machine M_{V^*} sets $\alpha = V^*(x, r; H_1, \psi_1, \ldots, H_{i-1}, \psi_{i-1}, H)$. ($\alpha$ is the message V^* sends on input x and internal coins r after receiving messages $H_1, \psi_1, \ldots, H_{i-1}, \psi_{i-1}, H$.) Without loss of generality, $\alpha \in \{0,1\}$ (since otherwise it is treated as $\alpha = 0$). There are two cases.

Case 1. $\alpha = \beta$ (lucky for M_{V^*}). Machine M_{V^*} appends (H, α, ψ) to its communication-record tape, sets $H_i \leftarrow H$, $\alpha_i \leftarrow \alpha$, $\psi_i \leftarrow \psi$, and proceeds to the next round (i.e., $i \leftarrow i + 1$).

Case 2. $\alpha \neq \beta$ (unlucky for M_{V^*}). Machine M_{V^*} is going to repeat the current round. Nothing is appended to the record tape, i is not increased, and steps (S1) and (S2) are repeated.

If all rounds are completed then M_{V^*} halts outputting (x, r', γ), where r' is the prefix of r scanned by V^* on input x, internal coin tosses r and communication record γ. We stress that M_{V^*} outputs the prefix of r determined by V^* actions in the conversation that is being output, and not the possibly longer prefix that M_{V^*} has used in previous attempts!

We now have to prove the validity of the construction. First, we prove that the simulator M_{V^*} indeed terminates in expected polynomial-time. Next, we prove that the output distribution produced by M_{V^*} equals the distribution over V^*'s tapes when interacting with P. Once these two claims are proven, the theorem follows.

CLAIM 2.1. *Machine M_{V^*} terminates in expected polynomial-time.*

PROOF. Let $\gamma_j = (H_j, \psi_j)$; and $\gamma^{(i)}$ denote $\gamma_1, \ldots, \gamma_i$. For each repetition of the $i+1$st round, β and H are statistically independent (since H is a random isomorphic copy of G_1 given either values of β). Given that $\beta \in_R \{0,1\}$, we get

$$\Pr\left(V^*(x, r; \gamma^{(i)}, H) = \beta\right) = \frac{1}{2}.$$

Thus, the expected number of times that M_{V^*} repeats each round is exactly 2. The claim follows from the fact that V^* itself is expected polynomial-time. □

CLAIM 2.2. *The probability distribution $M_{V^*}(G_1G_0)$ is identical to the distribution $\langle P, V^*\rangle(G_1G_0)$.*

PROOF. Let $x = (G_1, G_0)$. A typical element in the support of $M_{V^*}(x)$ (and $\langle P, V^*\rangle(x)$) consists of the input x, a random tape prefix r', and a sequence of m triples $(H_1, \alpha_1, \psi_1), \ldots, (H_m, \alpha_m, \psi_m)$. Both distributions depend in fact on r, the infinite sequence of coin tosses for the verifier, which in both cases is chosen uniformly. It suffices thus to show that for every fixed value of r the residual (conditional) distributions are equal. These residual distributions, depending on r, consist of a prefix r' and a sequence of m triples. Note that r', the finite prefix of r scanned by V^* is determined by r and the messages that V^* has received.

For any fixed infinite sequence r, and $0 \le i \le m$, let $\Pi_{PV^*}^{(x,r,i)}$ denote the probability distribution defined by the i first rounds of message exchange between P and V^* on common input x when V^* uses r as its source of internal coin tosses. Similarly, $\Pi_M^{(x,r,i)}$ denotes the probability distribution defined by the i first triples output by M_{V^*} on input x when fixing r as the verifier's source of internal coin tosses. The claim follows by proving that $\Pi_M^{(x,r,m)} = \Pi_{PV^*}^{(x,r,m)}$, and this equality is proven by induction on i.

Induction Base ($i = 0$). Hold vacuously.

Induction Step ($i \to i+1$). Let $\gamma^{(i)}$ be an element in the support of $\Pi_M^{(i)}$. By induction hypothesis

$$\Pr\left(\Pi_M^{(x,r,i)} = \gamma^{(i)}\right) = \Pr\left(\Pi_{PV^*}^{(x,r,i)} = \gamma^{(i)}\right). \tag{1}$$

Conditioning on the event

$$\Pi_M^{(x,r,i)} = \gamma^{(i)}$$
$$\left(\text{resp., } \Pi_{PV^*}^{(x,r,i)} = \gamma^{(i)}\right),$$

we consider the $i+1$st triple in $\Pi_M^{(x,r,i+1)}$ (resp., $\Pi_{PV^*}^{(x,r,i+1)}$), denoted $\Pi_M^{(i+1)}$ (resp., $\Pi_{PV^*}^{(i+1)}$). We show that both distributions are uniform over the set

$$S = \left\{(H, \alpha, \psi) : H = \psi G_\alpha \wedge \alpha = V^*\left(x, r, \gamma^{(i)}H\right)\right\}.$$

The set S is in 1–1 correspondence with the set $Sym(V)$. The correspondence is given by the mapping $(H, \alpha, \psi) \to \psi \cdot \phi^{\alpha-1}$. This mapping is 1–1 since collision implies equal H's (as $H = \psi G_\alpha = \psi \cdot \phi^{\alpha-1}G_1$), equal H's imply equal α's (as α is determined by H), and equal α's imply equal ψ's. The inverse mapping is $\pi \to (\pi G_1, \alpha(\pi), \pi \cdot \phi^{1-\alpha(\pi)})$, where $\alpha(\pi) = V^*(x, r, \gamma^{(i)}(\pi G_1))$. The inverse mapping coincides with the way an element

is chosen in the distribution $\Pi_{PV^*}^{(i+1)}$, and thus this distribution is uniform on S. Another way of sampling uniformly the set S is exhibited by $\Pi_{M_{V^*}}^{(i+1)}$: choosing uniformly $\beta \in \{0, 1\}$ and ψ in $Sym(V)$ and outputting $(\psi G_\beta, \beta, \psi)$ if $\beta = V^*(x, r, \gamma^{(i)}(\psi G_\beta))$. In case there is output, it is uniformly distributed in S. Thus,

$$\begin{aligned}\Pr\left(\Pi_M^{(x,r,i+1)} = \gamma^{(i)} H_{i+1}\alpha_{i+1}\pi_{i+1} \mid \Pi_M^{(i)} = \gamma^{(i)}\right) \\ = \Pr\left(\Pi_{PV^*}^{(x,r,i+1)} = \gamma^{(i)} H_{i+1}\alpha_{i+1}\pi_{i+1} \mid \Pi_{PV^*}^{(i)} = H^{(i)}\right). \qquad (2)\end{aligned}$$

Combining (1) and (2), the claim follows. □

Combining the two claims, the theorem follows. □

Remark 10. *Serial execution vs. parallel execution: the case where intuition fails?* Although one's intuition may insist that the above zero-knowledge protocol remains zero-knowledge even when the m rounds are executed in parallel instead of serially, we do not know how to prove this statement. In fact, recent work of Krawczyk and Goldreich [40] shows that the parallel version is unlikely to be zero-knowledge, since its alleged zero-knowledge property cannot be proven using a "black-box" simulation, unless $GI \in \text{BPP}$. Interestingly, a *modification* of the parallel version (in which the verifier uses "secret coins") yields a constant-round perfect zero-knowledge proof for GI [9]. Getting back to the proof of Theorem 2, the reader should note that the argument used in the proof is based on the fact that the verifier's request for the ith graph only depends on the first i graphs. Thus, changing the $i+1$st graph does not affect the requests for the previous i graphs. This may not be the case in a parallel execution of the protocol, where the prover first sends m isomorphic copies and only then receives the verifier's requests. In this case, the verifier's request for the ith graph may depend on all the other graphs. If we try to satisfy all the requests by choosing the H_i's beforehand, we have only a negligible probability of succeeding (i.e., 2^{-m}). If we try to construct a good sequence of H_i's, by choosing the H_i's at random and replacing the bad H_i's hoping that the new α_i's will match these graphs better, we are likely to fail as the α_i's corresponding to the good H_i's may change too.

Remark 11. It seems that allowing the simulator to be expected polynomial-time (rather than strictly probabilistic polynomial-time) is essential to the *perfect* zero-knowledge property of Protocol 2, even when restricting the verifier to be strictly probabilistic polynomial-time. In particular, the simulator presented in the proof of Theorem 2 has (rare) super-polynomial runs. An easy modification sacrifices the perfectness of the simulation to get strictly polynomial-time simulators for strictly polynomial-time verifiers. This modification consists of repeating each round in the simulation at most m times and stopping in the highly unlikely case that all m tries were unsuccessful. The resulting simulator has running time bounded by m times the running time of the verifier and produces a probability distribution with a statistical difference $m2^{-m}$ from the distribution of the interactions with the real prover. Thus, using "strictly polynomial-time" instead of "expected polynomial-time" in the definition of zero-knowledge, we get that Protocol 2 is almost-perfect zero-knowledge.

2.3 ZERO-KNOWLEDGE PROOF SYSTEM FOR GRAPH NONISOMORPHISM. The interactive proof system for graph nonisomorphism presented in Section 2.1 is probably not zero-knowledge: a user interacting with the prover may use the prover in order to discover which of the given graphs (G_1 and G_0) is isomorphic to a third graph G_3 (for more details, see [44] and [60]). The way to fix this flaw is to let the verifier first "prove" to the prover that he "knows" an isomorphism between his query graph H and one of the input graphs. This is done by using a parallel version of Protocol 2 in order to prove that H is isomorphic to either G_1 or G_0. We do not require this subprotocol to be zero-knowledge but rather to "hide" the isomorphism used in the proof. Formulating the appropriate definition of what it means for a machine to "prove knowledge of something" is quite complex and beyond the scope of this paper. We only note that the definitions proposed in [27], [47], and [67] do not suffice. We prefer to present our zero-knowledge proof system for *GNI*, and demonstrate its validity "directly", without using a notion of a "proof of knowledge".

The modification to Protocol 1 follows the underlying ideas of the zero-knowledge proof system of quadratic nonresiduosity that appeared in [47]. A unified and generalized presentation appears in [67]. The presentation of the protocol is further simplified using the "pairing trick" of Benaloh [10].[7]

Protocol 3

common input. Two graphs $G_1(V, E_1)$ and $G_0(V, E_0)$.

The following five steps are executed m times, each time using independent random coin tosses.

(V1) The verifier chooses at random $\alpha \in_R \{0, 1\}$, and a permutation $\pi \in_R Sym(V)$. The verifier constructs $H = \pi G_\alpha$. (The graph H is a random isomorphic copy of either G_1 or G_0.) The graph H will be called the *question*. In addition to the graph H, the verifier constructs n^2 pairs[8] of graphs such that each pair consists of one random isomorphic copy of G_1 and one random isomorphic copy of G_0. The graphs in each pair are placed at random order. These pairs will be called the *test pairs*, as they will be used by the prover to test whether the verifier is not cheating. Specifically, for each $1 \le i \le n^2$, the verifier chooses at random a bit $\gamma_i \in_R \{0, 1\}$ and two permutations $\tau_{i,1}, \tau_{i,0} \in_R Sym(V)$, and computes, for $j \in \{0, 1\}$, $T_{i,j} = \tau_{i,j} G_{j+\gamma_i \bmod 2}$. The verifier sends H and the sequence of pairs $(T_{1,1}, T_{1,0}), \ldots, (T_{n^2,1}, T_{n^2,0})$ to the prover.

(P1) The prover chooses at random, a subset $I \subseteq \{1, 2, \ldots, n^2\}$. ($I$ is chosen with uniform probability distribution among all 2^{n^2} subsets.) The prover sends I to the verifier.

(V2) If I is not a subset of $\{1, 2, \ldots, n^2\}$ then the verifier halts and *rejects*. Otherwise, the verifier replies with $\{(\gamma_i, \tau_{i,1}, \tau_{i,0}): i \in I\}$ and $\{(\alpha + \gamma_i \bmod 2, \tau_{i,(\alpha+\gamma_i \bmod 2)}\pi^{-1}): i \in \bar{I}\}$, where $\bar{I} = \{1, 2, \ldots, n^2\} - I$. Namely, for $i \in I$ the verifier shows an isomorphism between the input

[7] The underlying idea of this simplification appears also in [26].

[8] n^2 is an arbitrary choice of a function which is bounded above by a polynomial in n and bounded below by the logarithm of the complexity of testing isomorphism (between two graphs on n nodes). This choice is of course not optimal (especially in light of [7]).

graphs and the ith pair, while for $i \in \bar{I}$ the verifier shows an isomorphism between H and one of the graphs in the ith pair. Intuitively, for $i \in I$ the verifier shows that the ith pair is properly constructed, while for $i \in \bar{I}$ the verifier shows that if the ith pair was properly constructed then so was H.

(P2) The prover checks whether $\tau_{i,j}$ is indeed the isomorphism between $T_{i,j}$ and $G_{(j+\gamma_i \bmod 2)}$, for every $i \in I$ and $j \in \{0, 1\}$, and whether $\tau_{i,(\alpha+\gamma_i \bmod 2)}\pi^{-1}$ is indeed an isomorphism between $T_{i,(\alpha+\gamma_i \bmod 2)}$ and H, for every $i \in \bar{I}$. If either condition is violated, the prover stops. Otherwise, the prover answers with $\beta \in \{1, 0\}$, such that H is isomorphic to G_β. (If no such β exists, the prover answers with $\beta = 0$.)

(V3) The verifier tests whether $\alpha = \beta$. If the condition is violated, the verifier stops and *rejects*; otherwise, he continues.

If the verifier has completed m iterations of the above steps, then he *accepts*.

When verifying that the modified protocol still constitutes an interactive proof system for graph nonisomorphism, we have to show that the information revealed by the verifier in Step (V2) does not yield information about α (in case the graphs are isomorphic!). When proving that Protocol 3 is zero-knowledge, we use the information revealed in Step (V2) to help the simulator.

THEOREM 3. *Protocol 3 constitutes a (perfect) zero-knowledge interactive proof system for graph nonisomorphism.*

PROOF. First, we prove that Protocol 3 (which is a modification of Protocol 1) is still an interactive proof system for graph nonisomorphism. Clearly, the completeness condition still holds (i.e., if the graphs G_1 and G_0 are not isomorphic, and both prover and verifier follow the protocol, then the verifier always accepts). If, on the other hand, G_1 and G_0 are isomorphic then the graphs and permutations sent by the verifier in Steps (V1) and (V2) are random isomorphic copies of both graphs and yield no information about α. It follows that if G_1 and G_0 are isomorphic then $\alpha \neq \beta$ with probability at least $\frac{1}{2}$, even if the prover does not follow the protocol. Repeating the five steps m times we conclude that the probability that the verifier does not reject two isomorphic graphs is at most 2^{-m}.

We now turn to show that Protocol 3 is indeed zero-knowledge. Again, we use Definition 5. As in the proof of Theorem 2, we present a machine M_{V^*} for every interactive machine V^*, such that $M_{V^*}(x) = \langle P, V^* \rangle(x)$. The machine M_{V^*} uses V^* as a subroutine. The structure of the simulators we construct here is different from the simulators constructed in the proof of Theorem 2. In the proof of Theorem 2, the simulator produced conversations by guessing in advance what his subroutine V^* will ask. In this proof, the simulator will produce conversations by extracting from V^* the knowledge it has about its questions. In doing so, we implicitly use the notion of a "machine knowing something". (Since there is no explicit use of this notion, there is no need to present a definition here.)

Following is a detailed description of M_{V^*}. In this description, we assume that the interactive machine V^* is strictly polynomial-time. The modification to expected polynomial-time is along the lines discussed in the proof of Theorem 2. Machine M_{V^*} starts by choosing a random tape $r \in_R \{0, 1\}^q$ for V^*, where

q is a bound on the running time of V^* on input G_1G_0. M_{V^*} places r on its record tape and proceeds in m rounds as follows. (Initially $t = 1$.)

Round t:

(S1) M_{V^*} initiates V^* on input G_1G_0 and random tape r, and reads the graphs H, and $(T_{1,1}, T_{1,0}), \ldots, (T_{n^2,1}, T_{n^2,0})$ from the communication tape of V^* (i.e., $H, (T_{1,1}, T_{1,0}), \ldots, (T_{n^2,1}, T_{n^2,0}) \leftarrow V^*(G_1G_0, r)$).[9] Machine M_{V^*} chooses a random subset I and places it on the communication tape of V^*. M_{V^*} also appends I to its record tape.

(S2) M_{V^*} reads $\{(\gamma_i, \tau_{i,1}, \tau_{i,0}): i \in I\}$ and $\{(\alpha + \gamma_i \bmod 2, \tau_{i,(\alpha+\gamma_i \bmod 2)}\pi^{-1}): i \in \bar{I}\}$, from the communication tape of V^* (i.e., $\{(\gamma_i, \tau_{i,1}, \tau_{i,0}): i \in I\}$, $\{(\alpha + \gamma_i \bmod 2, \tau_{i,(\alpha+\gamma_i \bmod 2)}\pi^{-1}): i \in \bar{I}\} \leftarrow V^*(G_1G_0, r; I)$). Machine M_{V^*} checks that $\tau_{i,j}$ is an isomorphism between $T_{i,j}$ and $G_{(j+\gamma_i \bmod 2)}$, for every $i \in I$ and $j \in \{0, 1\}$. It also checks that $\tau_{i,(\alpha+\gamma_i \bmod 2)}\pi^{-1}$ is an isomorphism between $T_{i,(\alpha+\gamma_i \bmod 2)}$ and H, for every $i \in \bar{I}$. If either conditions is violated then M_{V^*} outputs its record tape and stops. Otherwise, M_{V^*} continues to Step (S3).

(S3) The purpose of this step is to find an isomorphism between H and G_α. This is done by repeating the following procedure (until such an isomorphism is found).

  (S3.1) Machine M_{V^*} chooses at random a subset $K \subseteq \{1, 2, \ldots, n^2\}$. Machine M_{V^*} initiates V^* on input G_1G_0, (the same!) random tape r, and places K as the first message on the read-only communication tape of V^*. Consequently, machine M_{V^*} reads $\{(\delta_i, \sigma_{i,1}, \sigma_{i,0}): i \in K\}$ and $\{(\alpha + \delta_i \bmod 2, \sigma_{i,(\alpha+\delta_i \bmod 2)}\pi^{-1}): i \in \bar{K}\}$, from the communication tape of V^*. (That is, $\{(\delta_i, \sigma_{i,1}, \sigma_{i,0}): i \in K\}$, $\{(\alpha + \delta_i \bmod 2, \sigma_{i,(\alpha+\delta_i \bmod 2)}\pi^{-1}): i \in \bar{K}\} \leftarrow V^*(G_1G_0, r; K)$.)

  (S3.2) For every $i \in I \cap \bar{K}$, if $\sigma_{i,(\alpha+\delta_i \bmod 2)}\pi^{-1}$ is indeed an isomorphism between $T_{i,(\alpha+\delta_i \bmod 2)}$ and H then $\tau^{-1}_{i,(\alpha+\delta_i \bmod 2)}\sigma_{i,(\alpha+\delta_i \bmod 2)}\pi^{-1}$ is an isomorphism between G_α and H. Analogously, for $i \in \bar{I} \cap K$. If an isomorphism was not found, go to Step (S3.1).

  (S3.3) In parallel, try to find an isomorphism between H and either input graphs by exhaustive search. One trial is done per each invocation of V^*. (In case M_{V^*} finds that H is not isomorphic to either input graphs, M_{V^*} acts as if it has found an isomorphism between H and G_0.)

(S4) Once an isomorphism between H and G_β ($\beta \in \{0, 1\}$) is found, machine M_{V^*} appends β to its record tape, thus completing round t.

If all rounds are completed then M_{V^*} outputs its record and halts.

The intuition behind the construction of the simulator M_{V^*} is that if V^* has ϵ probability of passing Step (P2) in the protocol then the same holds with

[9] The notation $V^*(G, G_0, r)$ is accurate only for the first round ($t = 1$). In subsequent rounds, V^* actions depend also on the transcript of the previous rounds; hence, $V^*(G, G_0, r)$ should be substituted by $V^*(G, G_0, r, H^{(t-1)})$, where $H^{(t-1)}$ is the transcript (conversation) generated for the previous $t - 1$ rounds. The notation $V^*(G, G_0, r)$ is suitable also for the parallel version of the protocol (discussed in Remark 12).

respect to M_{V^*} entering Step (S3). In this case, the simulator has to come up with the right answer concerning the isomorphism between H and one of the input graphs. To do so, the simulator will invoke V^* again with the same random tape (causing the verifier to produce the same question and the same test pairs) and ask him about a different subset K. The proper conclusion of Step (S2) has provided the simulator with isomorphisms between the test graph-pairs and the input graphs (for the tests with index in I). If V^* supplies an isomorphism between H and one of the test graphs (for a test with index $i \in \bar{K}$) and $i \in I$ then an isomorphism between H and one of the input graphs is constructed (by transitivity). Analogously for $i \in \bar{I} \cap K$. It all boils down to the question what is the probability of getting from V^* a good answer on a randomly chosen $K \neq I$. This probability is almost ϵ (the difference may be $\approx 2^{-n^2}$). Thus the expected number of times M_{V^*} repeats Step (S3) on the condition that it passes Step (S2) is $O(1/\epsilon)$, provided that $\epsilon > 2^{-n^2}$. This guarantees expected polynomial-time simulation for every value of $\epsilon \geq 2 \cdot 2^{-n^2}$. But what happens if $\epsilon = 2^{-n^2}$ (this may happen if V^* constructs his test pairs so that he can answer on a single subset (he can do so without knowing the isomorphism between H and either input graphs)). This extreme case is handled by Step (S3.3): the simulator finds an isomorphism by running an ordinary algorithm (e.g. exhaustive search). Note that this is done only in the unlikely event (occurring with probability 2^{-n^2}) that M_{V^*} passed Step (S2). Without Step (S3.3) the simulator would have produced almost the same distribution as in the prover-verifier conversations and hence Protocol 3 would have been "only" proven to be almost-perfect zero-knowledge.

We now prove the validity of the construction. First, we prove that the simulator M_{V^*} indeed terminates in expected polynomial-time. Next, we prove that the output distribution produced by M_{V^*} does equal the distribution over V^*'s tapes (when interacting with P). Once these two claims are proven, the theorem follows.

CLAIM 3.1. *Machine M_{V^*} terminates in expected polynomial-time.*

PROOF. We consider the expected running time on a single round, for any fixed random tape r. The reader can easily extend the proof to all rounds. We call a subset $I \subseteq \{1, 2, \ldots, n^2\}$ *good* if V^* answers properly on message I (i.e., I is good iff $V^*(G_1 G_0, r; I)$ consists of isomorphisms between the $T_{i,1}$, $T_{i,0}$ and G_1, G_0 for $\forall i \in I$ and an isomorphism between one of the $T_{i,j}$'s and H for $\forall i \in \bar{I}$). Denote by g_r the number of good subsets (g_r depends of course also on V^*, the input graphs and previous rounds which are all fixed here). Clearly, $0 \leq g_r \leq 2^{n^2}$. We compute the expected number of times V^* is invoked at round t as a function of g_r. We need to consider three cases.

Case 1 ($g_r \geq 2$). In case the subset I chosen in Step (S1) is good, we have to consider the probability that a randomly selected subset K is good and $\neq I$. In case the set chosen in Step (S1) is bad, the round is completed immediately. Thus, the expected number of invocations of V^* is

$$1 + \frac{g_r}{2^{n^2}} \cdot \left(\frac{g_r - 1}{2^{n^2}} \right)^{-1} = 1 + \frac{g_r}{g_r - 1} \leq 3.$$

Case 2 ($g_r = 1$). With exponentially small probability (i.e., 2^{-n^2}) the subset I chosen in Step (S1) is good. In this case, we find the isomorphism by

exhaustive search (i.e., $n!$ steps). Otherwise, the round is completed immediately. Thus, the expected complexity of M_{V^*} in Case 2 is bounded by one invocation of V^* and an additional $n! \cdot 2^{-n^2} < 1$ step.

Case 3 ($g_r = 0$). The subset I chosen in Step (S1) is always bad, and thus M_{V^*} invokes V^* exactly once.

The claim follows by the fact that V^* is polynomial-time. □

CLAIM 3.2. *The probability distribution $M_{V^*}(G_1G_0)$ is identical to the distribution $\langle P, V^* \rangle (G_1G_0)$.*

PROOF. Both distributions consist of a random r, and sequence of elements, each of which is either (I, β) (with good I and β so that H is isomorphic to G_β) or a bad I, where I is a random subset of $\{1, 2, \ldots, n^2\}$. □

The theorem follows. □

Remark 12. Protocol 3 was presented as a sequential application of m copies of a five-step protocol. However, the validity of Theorem 3 does not rely on the fact that these copies are executed sequentially, rather than in parallel. The five-step protocol in which copies of the five-step subprotocol (of Protocol 3) are performed in parallel is also a zero-knowledge interactive proof system for graph nonisomorphism. The simulator can handle all copies concurrently, by first by choosing a subset I_t for each copy t, and next choosing subsets K_t for each copy t. The fact that Protocol 3 can be parallelized follows from the fact that its proof technique only needs "something good" (choosing a good I_t) to happen once per each copy (and not necessarily concurrently!). In the proof of Theorem 2, we need "something good" (choosing a lucky β_t) to happen concurrently in all copies.

Remark 13. As in Theorem 2, it seems that allowing the simulator to be expected polynomial-time (rather than strictly probabilistic polynomial-time) is essential to the *perfect* zero-knowledge property of Protocol 3, even when restricting the verifier to be strictly probabilistic polynomial-time. However, the situation here is more acute. Bounding the number of iterations (of Step (S3)) of our simulator by a particular polynomial may result in an output with nonnegligible statistical difference from the distribution of the interactions with the real prover. (The statistical difference achieved by this strictly polynomial-time simulator can be guaranteed to be less than one over a related polynomial, but not less than any polynomial.) Thus, we do not know whether Protocol 3 is almost-perfect zero-knowledge, when using "strictly polynomial-time" instead of "expected polynomial-time" in the definition of zero-knowledge.

3. *All Languages in NP Have Zero-Knowledge Proof Systems*

In this section, we present zero-knowledge interactive proof systems for every language in NP, assuming the existence of secure encryption functions (see definition below). We begin by presenting a zero-knowledge interactive proof system for graph 3-colorability. Using this interactive proof system and the NP-Completeness of graph 3-colorability, we present zero-knowledge proof systems for every language in NP.

In this section, we assume the existence of *secure* encryption schemes. Security is defined (see below) analogously to the definitions of Goldwasser and

Micali [46]. However, the encryption schemes need not be "public-key." In fact, it suffices to have "bit commitment schemes" (which using recent results of Naor [58], Impagliazzo et al. [52], and Hastad [51] exist, assuming the existence of one-way functions).[10] A concrete and more efficient implementation (of secure encryption), assuming the intractability of factoring, appears in [4]. Here we only use encryptions for four messages; namely, 0, 1, 2, and 3. This allows a more specific presentation of the notions of an encryption scheme and its security.

Definition 6 (*Secure Encryption* [46]). An *encryption function* is a polynomial-time computable two-argument function $f\colon \{0,1,2,3\} \times \{0,1\}^* \to \{0,1\}^*$ satisfying $\forall x \neq y \in \{0,1,2,3\}$ and $\forall r, s \in \{0,1\}^*$ $f(x,r) \neq f(y,s)$.

A *probabilistic encryption* of a message x *with security parameter* n is a random variable $f_n(x) = f(x,r)$, where f is an encryption function and $r \in_R \{0,1\}^n$.

An encryption function f is *secure* if, for $\forall x \neq y \in \{0,1,2,3\}$, the ensembles $\{f_n(x)\}_n$ and $\{f_n(y)\}_n$ are polynomially indistinguishable.

An encryption function f is *nonuniformly secure* if the above ensembles are *indistinguishable by polynomial-size circuits*. Namely, for $\forall x \neq y \in \{0,1,2,3\}$, every polynomial Q and every family of Boolean circuits $\{C_n\}_n$, where C_n is a circuit of size $Q(n)$, every constant $c > 0$ and sufficiently large n

$$\left|\Pr\left(C_n(f_n(x)) = 1\right) - \Pr\left(C_n(f_n(y)) = 1\right)\right| < \frac{1}{n^c}.$$

3.1 A Zero-Knowledge Proof System for Graph 3-Colorability. A graph $G(V,E)$ is said to be 3-*colorable* if there exists a mapping $\phi\colon V \to \{1,2,3\}$ (called a *proper coloring*) such that every two adjacent vertices are assigned different colors (i.e., each $(u,v) \in E$ satisfies $\phi(u) \neq \phi(v)$). Such 3-coloring induces a partition of the vertex set of the graph to three independent sets. The language *graph* 3-*colorability*, denoted $G3C$, consists of the set of undirected graphs that are 3-*colorable*. *Graph* 3-*colorability is known to be NP-complete* [35].

The common input to the following protocol is a graph $G(V,E)$, which we assume without loss of generality to be simple and connected. In the following protocol, the prover needs only to be a probabilistic polynomial-time machine which gets a proper 3-coloring of G as an auxiliary input. Let us denote this coloring by ϕ ($\phi\colon V \to \{1,2,3\}$). Let $n = |V|$, $m = |E|$, and $S_3 = Sym(\{1,2,3\})$. Since the graph is (simple and) connected, n and m are polynomially related (i.e., $n - 1 \leq m < n^2/2$). For simplicity, let $V = \{1,2,\ldots,n\}$.

To clarify the presentation of our protocol, we first present an implementation of it which uses "physical means of hiding information." To be more specific, we assume that the prover has at its disposal a large supply of boxes each having a lock with a corresponding key. All keys are different.

[10] The notion of one-way function originates from [25]. A seemingly weaker, yet equivalent, formulation appears in [68]. Notable concrete suggestions are due to [61] and [63].

Protocol 4 (*Physical Implementation*)

common input. A graph $G(V, E)$.

The following four steps are executed m^2 times, each time using independent coin tosses.

(P1) The prover chooses at random an assignment of three colors to the three independent sets induced by ϕ, colors the graph using this 3-coloring, and places these colors in n locked boxes each bearing the number of the corresponding vertex. More specifically, the prover chooses a permutation $\pi \in_R S_3$, places $\pi(\phi(i))$ in a box marked i $(\forall i \in V)$, locks all boxes and sends them (without the keys) to the verifier.

(V1) The verifier chooses at random an edge $e \in_R E$ and sends it to the prover. (Intuitively, the verifier asks to examine the colors of the endpoints of $e \in E$.)

(P2) If $e = (u, v) \in E$, then the prover reveals the colors of u and v, by sending him the keys to boxes u and v. Otherwise, the prover does nothing.

(V2) The verifier opens boxes u and v using the keys received and checks whether they contain two different elements of $\{1, 2, 3\}$. If the keys do not match the boxes, or the contents violates the condition then the verifier *rejects* and stops. Otherwise, the verifier continues to the next iteration.

If the verifier has completed all m^2 iterations then it *accepts*.

It is easy to see that Protocol 4 constitutes an "interactive proof system" for Graph 3-Colorability. If the graph is 3-colorable and both prover and verifier follow the protocol, then the verifier accepts with probability 1. If the graph is not 3-colorable and the verifier follows the protocol then, no matter how the prover plays, at each round the verifier will reject with probability at least $1/m$. It follows that the probability that the verifier will accept (i.e., complete all the m^2 rounds without detecting that "something is wrong") is bounded above by $(1 - m^{-1})^{m^2} \approx \exp(-m)$.

Loosely speaking, Protocol 4 is "zero-knowledge" since the only information received by the verifier at each round is a pair of different randomly selected elements of $\{1, 2, 3\}$. It is crucial that the prover uses at each round an independently selected random permutation of the colors. Thus, the names of the three classes (of the partition ϕ) at one round are uncorrelated to the names at another round. It follows that the values of every random permutation at the end-points of one edge reveal no information about the coloring ϕ.

In the above paragraphs, the terms *interactive proof* and *zero-knowledge* were used in quotes since these were not defined for a physical setting. The adaptation is straightforward and is left to the reader.

We now present the digital implementation of Protocol 4. In this protocol, a standard (polynomial-time) algorithm for evaluating f is part of the programs of both prover and verifier.

Protocol 4

common input. A graph $G(V, E)$.

The following four steps are executed m^2 times, each time using independent coin tosses.

(P1) The prover chooses a random renaming of the 3-coloring, encrypts it, and sends the encryption to the verifier. More specifically, the prover chooses a permutation $\pi \in_R S_3$, and random n-bit r_v's ($\forall v \in V$, let $r_v \in_R \{0,1\}^n$). The prover computes $F_v = f(\pi(\phi(v)), r_v)$ (for every $v \in V$), and sends the sequence $F_1, F_2, \ldots, F_n$ to the verifier.

(V1) The verifier chooses at random an edge $e \in_R E$ and sends it to the prover.

(P2) If $e = (u, v) \in E$, then the prover reveals the coloring of u and v, and "proves" that they correspond to their encryptions. More specifically, the prover sends $(\pi(\phi(u)), r_u)$ and $(\pi(\phi(v)), r_v)$ to the verifier. If $e \notin E$, then the prover acts as if $e = e_0$, where e_0 is an arbitrary fixed edge in E.[11]

(V2) The verifier checks the "proof" provided in Step (P2). Namely, the verifier checks whether $F_u = f(\pi(\phi(u)), r_u)$, $F_v = f(\pi(\phi(v)), r_v)$, $\pi(\phi(u)) \neq \pi(\phi(v))$, and $\pi(\phi(u)), \pi(\phi(v)) \in \{1, 2, 3\}$. If either condition is violated the verifier *rejects* and stops. Otherwise, the verifier continues to the next iteration.

If the verifier has completed all m^2 iterations, then it *accepts*.

PROPOSITION 4. *If f is a nonuniformly secure encryption function, then Protocol 4 constitutes a zero-knowledge interactive proof system for Graph 3-Colorability.*

PROOF. First, we show that Protocol 4 constitutes an interactive proof system for $G3C$. The argument is as in the paragraph following the physical implementation of Protocol 4, except that it is important to stress that also in the digital implementation the prover cannot change the information hidden in his messages. This is due to the fact that each encrypted message has a unique decryption.

Our demonstration that Protocol 4 is indeed zero-knowledge follows the lines of the proof of Theorem 2. Namely, the construction of M_{V^*} uses the same underlying ideas used in the proof of Theorem 2. However, additional technical difficulties arise in proving the validity of the construction. The source of these difficulties is the fact that a cheating verifier may select his question based on the encryption of the coloring presented to him. The security of the encryption is carefully used to show that even doing so does not provide him any noticeable advantage.

As in the proof of Theorem 2, we present a machine M_{V^*} for every interactive machine V^*. This time, $M_{V^*}(x)$ and $\langle P, V^* \rangle(x)$ will not be equal, but instead it will be shown that they are polynomially indistinguishable. Typically, we try to guess which edge the machine V^* will ask to check. We encrypt an illegal coloring of G such that we can answer V^* in case we are lucky (i.e., V^* asks for the edge we chose). The cases in which we failed will be ignored. It is crucial that from the point of view of V^* the case that leads to our success and the case that leads to our failure are polynomially indistinguishable.

Again, we use Definition 5, including in the distributions (for sake of convenience) also the contents of the verifier's write-only communication tape.

[11] Here, as in Protocol 3, we have again preferred this "unnatural" convention that simplifies the zero-knowledge argument.

As in the proof of Theorem 3, we consider only probabilistic interactive machines that are strictly polynomial-time. The argument is easily extended to interactive machines that are expected polynomial-time (see proof of Theorem 2).

Following is a detailed description of M_{V^*}. The machine M_{V^*} uses V^* as a subroutine. Machine M_{V^*} starts by choosing a random tape $r \in_R \{0,1\}^q$ for V^*, where q is a bound on the running time of V^* on input G. Machine M_{V^*} places r on its record tape and proceeds in m^2 rounds as follows.

Round i:

(S1) M_{V^*} picks an edge $(u, v) \in_R E$ and a pair of integers $(a, b) \in_R \{(i, j): i \neq j \in \{1,2,3\}\}$ at random. M_{V^*} chooses random r_i's ($r_i \in_R \{0,1\}^n$) and computes $F_i = f(c_i, r_i)$ for each $i \in V$, where $c_i = 0$ for $i \in V - \{u, v\}$, $c_u = a$ and $c_v = b$. Machine M_{V^*} sets $R \leftarrow (F_1, F_2, \ldots, F_n)$.

(S2) Machine M_{V^*} sets $e = V^*(G, r; R_1, R'_1, \ldots, R_{i-1}, R'_{i-1}, R)$. ($e$ is the message that V^* sends on input G and random tape r after receiving messages $R_1, R'_1, \ldots, R_{i-1}, R'_{i-1}, R$, where R_j is the jth sequence of encrypted colors, R is the current sequence of encrypted colors, and R'_j is the jth revealed encryption). Without loss of generality, $e \in E$ (otherwise, e is treated as e_0—the fixed edge used in the protocol). We consider two cases.

Case 1. $e = (u, v)$ (lucky for M_{V^*}). Machine M_{V^*} sets $R_i \leftarrow R$, $e_i \leftarrow e$, $R'_i \leftarrow ((c_u, r_u), (c_v, r_v))$, appends (R_i, e_i, R'_i) to its record tape, and proceeds to the next round (i.e., $i \leftarrow i + 1$).

Case 2. $e \neq (u, v)$ (unlucky for M_{V^*}). Machine M_{V^*} is going to repeat the current round. Nothing is appended to the record tape, i is not increased, and Steps (S1) and (S2) are repeated.

When all m^2 rounds are completed, machine M_{V^*} outputs its record and halts.

We now have to prove the validity of the construction. First, we prove that the simulator M_{V^*} terminates in expected polynomial-time. This amounts to showing that, each time a round is repeated, the verifier's request (i.e., e) equals the chosen (u, v) with probability essentially $1/m$ (otherwise, V^* can be transformed into a circuit family that "breaks" the encryption function f). Next, we prove that the output distribution produced by M_{V^*} is polynomially indistinguishable from the distribution over V^*'s tapes when interacting with P. (There is clearly a difference between these probability distributions, but this difference cannot be "detected" in probabilistic polynomial-time.) Once these two claims are proven, the proposition follows. We start with the following technical lemma that asserts, as a special case, that the encryption of the coloring sent by the prover and the encryption of the "garbage" sent by the simulator are indistinguishable by polynomial-size circuits. The lemma is a special case of a theorem by Goldwasser and Micali [46], and its proof is sketched here for the sake of self-containment.

Notation. Let $\alpha = (\alpha_1, \ldots, \alpha_k)$, where $\alpha_i \in \{0,1,2,3\}$ for all $1 \leq i \leq k$. Then $\bar{f}_n(\alpha)$ denotes $f_n(\alpha_1) \cdots f_n(\alpha_k)$.

LEMMA 4.0. *Let P be a polynomial, and $\alpha^{(1)}, \alpha^{(2)}, \alpha^{(3)}, \ldots$ and $\beta^{(1)}, \beta^{(2)}, \beta^{(3)}, \ldots$ be two infinite sequences such that $\alpha^{(n)} = (\alpha_1^{(n)}, \ldots, \alpha_{k_n}^{(n)})$ and $\beta^{(n)} = (\beta_1^{(n)}, \ldots, \beta_{k_n}^{(n)})$, where $\alpha_i^{(n)}, \beta_i^{(n)} \in \{0, 1, 2, 3\}$ and $k_n < P(n)$. Then for every polynomial Q, every family of (bounded size) Boolean circuits $\{C_n: size(C_n) \leq Q(n)\}_n$, every constant $c > 0$ and sufficiently large n*

$$\left|Pr\left(C_n\left(\bar{f}_n(\alpha^{(n)})\right) = 1\right) - Pr\left(C_n\left(\bar{f}_n(\beta^{(n)})\right) = 1\right)\right| < \frac{1}{n^c}.$$

PROOF SKETCH. The statement is proven by considering encryptions of hybrids of prefixes of $\alpha^{(n)}$ and suffixes of $\beta^{(n)}$. A polynomial-size circuit family that distinguishes the encryptions of the extreme hybrids (i.e., $\alpha^{(n)}$ and $\beta^{(n)}$) also distinguishes encryptions of two adjacent hybrids. Incorporating the hybrid messages into the circuits one can construct a family of (polynomial-size) circuits that distinguishes the encryptions of $x \neq y \in \{0, 1, 2, 3\}$. This contradicts our assumption that f is a nonuniformly secure encryption function. □

Notations. For $\phi: \{1, 2, \ldots, n\} \to \{1, 2, 3\}$ and $\pi \in S_3$, denote by $msg_{PV}^{\phi,\pi}$ the sequence $(\pi(\phi(1)), \pi(\phi(2)), \ldots, \pi(\phi(n)))$. For $u, v \in \{1, 2, \ldots, n\}$ and $a, b \in \{1, 2, 3\}$, denote by $msg_M^{(u,v),(a,b)}$ the sequence $(c_1, c_2, \ldots, c_n)$, where $c_u = a$, $c_v = b$, and $c_i = 0$ for $i \in \{1, 2, \ldots, n\} - \{u, v\}$.

CLAIM 4.1. *Machine M_{V^*} terminates in expected polynomial time.*

PROOF. Let $R^{(i-1)}$ denote $R_1, R_1', \ldots, R_{i-1}, R_{i-1}'$, where R_j and R_j' are as in Step (S2) above. In each repetition of the ith round, machine M_{V^*} places a probabilistic encryption of $msg_M^{(u,v),(a,b)}$ on the communication tape of V^*, where $(u, v) \in_R E$ and $a \neq b \in_R \{1, 2, 3\}$. Using Lemma 4.0, we show that the probability that V^* asks to reveal the colors of u and v when seeing the encryption of $msg_M^{(u,v),(a,b)}$ on its communication tape approximately equals the probability V^* asks so when seeing the encryption of $msg_M^{(r,s),(a,b)}$ on its communication tape, $\forall (r, s) \in E$. It follows that a repetition of round i is completed successfully with probability $\approx 1/m$. A detailed proof follows.

We use $V^*(enc)$ as a shorthand for $V^*(G, r; R^{(i-1)}, enc)$. First, we bound the probability that round i is completed successfully. This probability equals the probability that $V^*(\bar{f}_n(msg_M^{(u,v),(a,b)})) = (u, v)$, where $(u, v) \in_R E$ and $a \neq b \in_R \{1, 2, 3\}$.

$$\begin{aligned}\Pr_{(u,v)\in_R E, a\neq b\in_R\{1,2,3\}}&\left(V^*\left(\bar{f}_n\left(msg_M^{(u,v),(a,b)}\right)\right) = (u, v)\right)\\ &= \sum_{(u,v)\in E} \frac{1}{m} \cdot \Pr_{a\neq b\in_R\{1,2,3\}}\left(V^*\left(\bar{f}_n\left(msg_M^{(u,v),(a,b)}\right)\right) = (u, v)\right).\end{aligned}$$

Let $(r, s) \in E$. Then by Lemma 4.0, V^* has on input $\bar{f}_n(msg_M^{(u,v),(a,b)})$ approximately the same output as on input $\bar{f}_n(msg_M^{(r,s),(a,b)})$. Otherwise, V^* and $R^{(i-1)}$ can be incorporated into a circuit that distinguishes the encryptions of $msg_M^{(u,v),(a,b)}$ and $msg_M^{(r,s),(a,b)}$ (for some u, v, r, s, a, b). It follows that the

above probability is bounded above by

$$
\begin{aligned}
&> \sum_{(u,v)\in E} \frac{1}{m} \cdot \left(\Pr_{a\neq b\in_R\{1,2,3\}} \left(V^*\left(\bar{f}_n\left(msg_M^{(r,s),(a,b)}\right)\right) = (u,v)\right) - \frac{1}{n^2} \right) \\
&> \frac{1}{m} \cdot \sum_{(u,v)\in E} \left(\Pr_{a\neq b\in_R\{1,2,3\}} \left(V^*\left(\bar{f}_n\left(msg_M^{(r,s),(a,b)}\right)\right) = (u,v)\right) - \frac{1}{2m} \right) \\
&= \frac{1}{m} \cdot \left(\Pr_{a\neq b\in_R\{1,2,3\}} \left(V^*\left(\bar{f}_n\left(msg_M^{(r,s),(a,b)}\right)\right) \in E\right) - \frac{1}{2} \right) \\
&= \frac{1}{2m}.
\end{aligned}
$$

(In the second inequality we use the fact that $m < n^2/2$.) We conclude that the expected number of times that each round is repeated is bounded below by $2m$. The claim follows by the hypothesis that V^* is polynomial-time. □

CLAIM 4.2. *The probability ensembles* $\{M_{V^*}(G)\}_{G\in G3C}$ *and* $\{\langle P, V^*\rangle(G)\}_{G\in G3C}$ *are polynomially indistinguishable.*

PROOF. The proof is by contradiction, and is first sketched in this paragraph. We assume that the two probability ensembles can be told apart by a family of polynomial-size circuits and derive a contradiction to Lemma 4.0 in one of essentially two ways. First, we restrict our attention to a single round of the $\langle P, V^*\rangle$ conversation and its simulation. Next, we consider two cases. If the distribution of the "verifier's query" (i.e., the edge e) is substantially different in the conversation and in the simulation, then the program V^* can be directly used to contradict Lemma 4.0. If these two distributions are close enough, we modify the circuits guaranteed by the above hypothesis to derive a contradiction to Lemma 4.0. The reader should note that the circuits in the hypothesis receive as input a text that is *not* of the form $\bar{f}_n(\alpha)$, for a sequence α, but rather a text consisting of such a sequence and two elements used for the randomization of the encryption. The location of these elements in the sequence may be determined by the entire sequence. This creates difficulties that need to be resolved with some care.

Let $\{C_n\}_n$ be a circuit family of polynomial size that distinguishes $\{M_{V^*}(G)\}_{G\in G3C}$ from $\{\langle P, V^*\rangle(G)\}_{G\in G3C}$. Namely, there exist a polynomial Q and a constant $c > 0$ such that the size of C_n is less than $Q(n)$ and an infinite sequence of graphs $G_i(V_i, E_i)$ such that for every i

$$
\Pr\left(C_{|V_i|}\left(M_{V^*}(G_i)\right) = 1\right) - \Pr\left(C_{|V_i|}\left(\langle P, V^*\rangle(G_i)\right) = 1\right) > \frac{1}{|V_i|^c}.
$$

Let us consider this circuit family. Without loss of generality, we assume that each circuit in the family has a single output bit. For sake of simplicity, we carry out our argument while referring to an arbitrary graph, denoted $G(V, E)$, in the above sequence of graphs and to the corresponding circuit $C_{|V|}$. It is important to note that all polynomials mentioned in the rest of the argument are fixed for the entire sequence and are independent of the generic graph G. Let ϕ denote the fixed coloring of $G(V, E)$ used by the prover, $n = |V|$ and $m = |E|$.

The next two paragraphs are devoted to show that it suffices to consider a simulation of a single round. The argument follows the "hybrid" technique first used in [46].

A typical element in the support of $M_{V^*}(G)$ (resp. in the support of $\langle P, V^*\rangle(G)$) consists of the input graph G, the random tape $r \in \{0,1\}^q$, and a sequence of m^2 triples $(R_1, e_1, R'_1), \ldots, (R_{m^2}, e_{m^2}, R'_{m^2})$. For $0 \le i \le m^2$, let $\Pi^{(i)}$ denote the ith hybrid distribution defined by taking G, $r \in \{0,1\}^q$ and the first i triples $(R_1, e_1, R'_1), \ldots, (R_i, e_i, R'_i)$ from the distribution $\langle P, V^*\rangle(G)$, and producing the remaining $m^2 - i$ triples by running M_{V^*} starting at the $i+1$st round using the random tape r and the history $R_1, R'_1, \ldots, R_i, R'_i$. Note that $\Pi^{(0)}$ equals $M_{V^*}(G)$, while $\Pi^{(m^2)}$ equals $\langle P, V^*\rangle(G)$.

Clearly, the circuit C_n that distinguishes the extreme hybrids (i.e., $\Pi^{(0)}$ and $\Pi^{(m^2)}$) also distinguishes two neighboring hybrids (i.e., $\Pi^{(i)}$ and $\Pi^{(i+1)}$, for some $i \in \{0, 1, \ldots, m^2 - 1\}$). Namely, there exists an i such that

$$\Pr\left(C_n\left(\Pi^{(i)}\right) = 1\right) - \Pr\left(C_n\left(\Pi^{(i+1)}\right) = 1\right) > \frac{1}{m^2 \cdot n^c}.$$

Let $\epsilon = 1/(m^2 \cdot n^c)$. Note that ϵ^{-1} as well as m/ϵ are polynomials in n (since $m \ge n - 1$).

Let $\Pi^{(i,j)}$ be the distribution obtained from $\Pi^{(i)}$ by omitting the last $m^2 - j$ triples. It is easy to see that this circuit family can be modified, maintaining the "distinguishing gap" (i.e., ϵ) and the polynomial upper bound on the size of its members, so that it distinguishes $\Pi^{(i,i+1)}$ and $\Pi^{(i+1,i+1)}$. (This is done by incorporating M_{V^*} into the circuits and applying it to construct the last $m^2 - (i+1)$ pairs. We get a probabilistic circuit that can be converted into a deterministic one while maintaining the "distinguishing gap".)

Let us take a closer look at $\Pi^{(i,i+1)}$ and $\Pi^{(i+1,i+1)}$. Intuitively, $\Pi^{(i+1,i+1)}$ is a distribution obtained by taking the first $i+1$ triples of $\langle P, V^*\rangle(G)$, while $\Pi^{(i,i+1)}$ is a distribution obtained by taking the first i triples of $\langle P, V^*\rangle(G)$ and letting M_{V^*} produce the $i+1$st triple. Let $\Pi_M(H_i)$ be a random variable such that $\Pi^{(i,i+1)} = H_i \Pi_M(H_i)$, where H_i is chosen according to $\Pi^{(i,i)}$. Similarly, let $\Pi_{PV}(H_i)$ be a random variable such that $\Pi^{(i+1,i+1)} = H_i \Pi_{PV}(H_i)$, where again H_i is chosen according to $\Pi^{(i,i)}$. Intuitively, $\Pi_M(H_i)$ (resp., $\Pi_{PV}(H_i)$) denotes the $i+1$st triple produced by M_{V^*} (resp., $\langle P, V^*\rangle$) when V^*'s input and communication tapes currently consists of the "history" H_i. Since the circuit C_n distinguishes $\Pi^{(i,i+1)}$ from $\Pi^{(i+1,i+1)}$ it follows that there *exists* an H_i such that the circuit C_n distinguishes (equally well) $H_i\Pi_M(H_i)$ from $H_i\Pi_{PV}(H_i)$. (This was just an averaging argument.) Incorporating this H_i into the circuit C_n, we get a circuit that distinguishes $\Pi_M(H_i)$ from $\Pi_{PV}(H_i)$. Namely,

$$\Pr\left(C_n\left(\Pi_M(H_i)\right) = 1\right) - \Pr\left(C_n\left(\Pi_{PV}(H_i)\right) = 1\right) > \epsilon \left(= \frac{1}{m^2 \cdot n^c}\right).$$

Recall that $\phi\colon \{1, 2, \ldots, n\} \to \{1, 2, 3\}$ is a fixed coloring of G. Note that $\Pi_{PV}(H_i)$ equals $(f_n(msg_{PV}^{\phi,\pi}), (u, v), (\pi(\phi(u)), r_u), (\pi(\phi(u)), r_v))$, where $\pi \in_R S_3$, $(u, v) = V^*(G, r; H_i, f_n(msg_{PV}^{\phi,\pi}))$ and r_u (resp., r_v) is the randomization used to encrypt the uth (resp., vth) element in $msg_{PV}^{\phi,\pi}$. On the

other hand, $\Pi_M(H_i)$ equals $(\bar{f}_n(msg_M^{(u,v),(a,b)}), (u, v), (a, r_u), (b, r_v))$, where u, v, a, b are chosen by some distribution, $(u, v) = V^*(G, r; H_i, \bar{f}_n(msg_M^{(u,v),(a,b)}))$ and r_u (resp., r_v) is the randomization used to encrypt the uth (resp., vth) element in $msg_M^{(u,v),(a,b)}$. A typical element in both distributions is an encryption of an n-element vector and two pairs corresponding to the decryption of two of its entries (the uth and vth entry).

We derive a contradiction (to Lemma 4.0) by using the circuit C_n to construct a circuit C'_n of polynomial-size, that distinguishes the encryptions of the following two messages msg_1 and msg_2, each being a sequence of $3n$ elements of $\{0, 1, 2, 3\}$. The sequence msg_1 consists of $3n$ *zero*s, while the sequence msg_2 consists of a sequence of n *one*s followed by a sequence of n *two*s and a sequence of n *three*s (i.e., $msg_1 = 0^{3n}$ and $msg_2 = 1^n2^n3^n$).

The Construction of C'_n. The circuit C'_n, incorporating the circuit C_n, the coloring ϕ, the (verifier's) program V^* and the history H_i, operates as follows. On input a text $t_1t_2 \cdots t_{3n}$, where $t_i \in \{0, 1\}^*$, the circuit C'_n selects at random $(u, v) \in_R E$, $\pi \in_R S_3$ and $r_u, r_v \in_R \{0, 1\}^n$. For $j \in \{u, v\}$, the circuit C'_n sets $c_j \leftarrow \pi(\phi(j))$ and $i_j \leftarrow (c_j - 1) \cdot n + j$ and constructs

$$text \leftarrow t_{i_1}t_{i_2} \cdots t_{i_{u-1}}f(c_u, r_u)t_{i_{u+1}} \cdots t_{i_{v-1}}f(c_v, r_v)t_{i_{v+1}} \cdots t_{i_n}.$$

(For $j \in \{1, 2, \ldots, n\} - \{u, v\}$, the jth element in *text* is the jth element in the $\pi(\phi(j))$th third of the input. The uth element in *text* is $f(c_u, r_u)$ and the vth element in *text* is $f(c_v, r_v)$.) The circuit C'_n then runs V^* on H_i and *text*. If $V^*(H_i, text) \neq (u, v)$, then C'_n stops outputting 0. If $V^*(H_i, text) = (u, v)$, then C'_n uses the circuit C_n and stops outputting $C_n(text, (u, v), (c_u, r_u), (c_v, r_v)) \in \{0, 1\}$. □

We are interested in the behavior of the circuit C'_n on the input distributions $\bar{f}_n(msg_1)$ and $\bar{f}_n(msg_2)$. On input distribution $\bar{f}_n(msg_1)$, the circuit C'_n runs V^* on input distribution $\bar{f}_n(msg_M^{(u,v),(a,b)})$, for $(u, v) \in_R E$ and $a \neq b \in_R \{1, 2, 3\}$. On input distribution $\bar{f}_n(msg_2)$, the circuit C'_n runs V^* on input distribution $\bar{f}_n(msg_{PV}^{\phi,\pi})$, for $\pi \in_R S_3$, We consider two cases. The easy case is the one where the distribution of V^*'s question (i.e., the edge it asks to reveal) is substantially different on the two input distributions. In this case V^* constitutes a contradiction to Lemma 4.0. The second case is when the distribution on V^*'s question is indistinguishable (on the two input distributions) so that the circuit C'_n runs the circuit C_n with probability $\approx 1/m$, on both input distributions. In this case, the circuit C'_n will distinguish the inputs relying on the distinguishing gap of C_n. A detailed argument follows.

Let D_1 (resp., D_2) denote the probability distribution of *text*, constructed by C'_n, on input distribution $\bar{f}_n(msg_1)$(resp., $\bar{f}_n(msg_2)$). The distribution D_1 (resp., D_2) depends on (u, v), π, r_u, r_v chosen by C'_n. Fixing such a choice, we get a distribution $D_1^{(u,v),\pi,r_u,r_v}$ (resp., $D_2^{(u,v),\pi,r_u,r_v}$), which is determined by the input distribution $\bar{f}_n(msg_1)$ (resp., $\bar{f}_n(msg_2)$). The distribution $D_1^{(u,v),\pi,r_u,r_v}$ consists of $u - 1$ encryptions of 0 followed by $f(\pi(\phi(u), r_u))$, another $v - u - 1$ encryptions of 0, followed by $f(\pi(\phi(v), r_v))$ and $n - v$ additional encryptions of 0. The distribution $D_2^{(u,v),\pi,r_u,r_v}$ consists of the concatenation of the distribution $f_n(\pi(\phi(1))), \ldots, f_n(\pi(\phi(u - 1)))$, the element $f(\pi(\phi(u), r_u))$, the distribution $f_n(\pi(\phi(u + 1))), \ldots, f_n(\pi(\phi(v - 1)))$, the element $f(\pi(\phi(v), r_v))$, and the distribution $f_n(\pi(\phi(v + 1)))$,

$\ldots, f_n(\pi(\phi(n)))$. We first prove that V^*, invoked by the circuit, has essentially the same output distribution, regardless of whether the input distribution is D_1 or D_2.

CLAIM 4.2.1

$$\left| Pr\left(V^*\left(H_t, D_1^{(u,v),\pi,r_u,r_v}\right) = (u, v)\right) - Pr\left(V^*\left(H_t, D_2^{(u,v),\pi,r_u,r_v}\right) = (u, v)\right)\right| < \frac{\epsilon}{2m},$$

where the probability is taken over all choices of $(u, v) \in E$, $\pi \in S_3$, *and* r_u, $r_v \in \{0, 1\}^n$, *with uniform probability distribution.*

PROOF. Assuming on the contrary that Claim 4.2.1 does not hold, using an averaging argument, we infer that there exist (fixed) u, v, and π such that V^* (with auxiliary input H_t) distinguishes $D_1^{(u,v),\pi,r_u,r_v}$ from $D_2^{(u,v),\pi,r_u,r_v}$, for r_u, $r_v \in_R \{0, 1\}^*$. Noting that these distributions are probabilistic encryptions of two n-element sequences, we derive a contradiction to Lemma 4.0 (by constructing a circuit incorporating V^* and H_i). □

We are now ready to show that C'_n distinguishes the encryption of msg_1 from that of msg_2.

CLAIM 4.2.2

$$\left| Pr\left(C'_n\left(\bar{f}(msg_1)\right) = 1\right) - Pr\left(C'_n\left(\bar{f}(msg_2)\right) = 1\right)\right| > \frac{\epsilon}{2m}.$$

PROOF. Let ξ be a random variable uniformly distributed upon $E \times S_3 \times \{0, 1\}^n \times \{0, 1\}^n$. We use D_1^ξ (resp., D_2^ξ) as a shorthand of $D_1^{(u,v),\pi,r_u,r_v}$ (resp., $D_2^{(u,v),\pi,r_u,r_v}$).

By the construction of C'_n, we have

$$\Pr\left(C'_n\left(\bar{f}(msg_1)\right) = 1\right) = \Pr\left(V^*\left(H_t, D_1^\xi\right) = (u, v)\right) \cdot \Pr\left(C_n\left(D_1^\xi, \xi'\right) = 1 \mid V^*\left(H_t, D_1^\xi\right) = (u, v)\right), \quad (1)$$

where $\xi = ((u, v), \pi, r_u, r_v) \in_R E \times S_3 \times \{0, 1\}^n \times \{0, 1\}^n$ and $\xi' = (u, v), (\pi(\phi(u)), r_u), (\pi(\phi(v)), r_v)$. Similarly,

$$\Pr\left(C'_n\left(\bar{f}(msg_2)\right) = 1\right) = \Pr\left(V^*\left(H_t, D_2^\xi\right) = (u, v)\right) \cdot \Pr\left(C_n\left(D_2^\xi, \xi'\right) = 1 \mid V^*\left(H_t, D_2^\xi\right) = (u, v)\right). \quad (2)$$

By Claim 2.1, we have

$$\left|\Pr\left(V^*\left(H_t, D_1^\xi\right) = (u, v)\right) - \Pr\left(V^*\left(H_t, D_2^\xi\right) = (u, v)\right)\right| < \frac{\epsilon}{2m}. \quad (3)$$

By observing that $D_2^{(u,v),\pi,r_u,r_v}$ is independent of the choice of $(u, v) \in E$ as long as $\pi \in_R S_3$ and $r_u, r_v \in_R \{0, 1\}^n$, we get

$$\Pr\left(V^*\left(H_t, D_2^\xi\right) = (u, v)\right) = \frac{1}{m} \quad (4)$$

and using the structure of $\langle P, V^* \rangle$ we get

$$\Pr\left(C_n\left(D_2^{\xi}, \xi'\right) = 1 \mid V^*\left(H_i, D_2^{\xi}\right) = (u, v)\right) = \Pr\left(C_n\left(\Pi_{PV}(H_i)\right) = 1\right). \quad (5)$$

By the construction of the simulator, M_{V^*}, we have

$$\Pr\left(C_n\left(D_1^{\xi}, \xi'\right) = 1 \mid V^*\left(H_i, D_1^{\xi}\right) = (u, v)\right) = \Pr\left(C_n\left(\Pi_M(H_i)\right) = 1\right). \quad (6)$$

Combining (1–6) we get

$$\left|\Pr\left(C_n'\left(\bar{f}(msg_1)\right) = 1\right) - \Pr\left(C_n'\left(\bar{f}(msg_2)\right) = 1\right)\right| > \frac{1}{m} \cdot \Delta - \frac{\epsilon}{2m},$$

where Δ is the absolute value of the difference between $\Pr(C_n(\Pi_M(H_i)) = 1)$ and $\Pr(C_n(\Pi_{PV}(H_i)) = 1)$.

Recalling that $\Delta > \epsilon$, Claim 2.2 follows. □

Once proved, Claim 2.2 contradicts Lemma 4.0, and thus our hypothesis that $\{M_{V^*}(G)\}_{G \in G3C}$ and $\{\langle P, V^* \rangle(G)\}_{G \in G3C}$ are polynomially distinguishable is contradicted. Claim 4.2 follows. □

Combining Claims 1 and 2, the Proposition follows. □

Remark 14. The above protocol needs m^2 rounds. There are two alternative ways of modifying the above protocol so to get a constant-round zero-knowledge protocol for graph 3-colorability. In both modifications, the idea is to have the verifier commit himself to all his queries (i.e., which edge he wants to check for each copy of the colored graph) before the prover sends to the verifier the corresponding colored graphs. The two modifications differ by the manner in which the verifier commits to his queries. One modification is based on the intractability of factoring or more generally on the existence of clawfree pairs of one-way permutations (see [48] for definition). The second modification is based on a relaxation of the definition of a proof system so that the prover is also restricted to polynomial-time (and his "computational advantage" over the verifier consists of an auxiliary input). This relaxation, referred to as an argument (see Section 1.4), is natural in the cryptographic applications. The details of the first modification are currently being worked out by Oded Goldreich and Ariel Kahn. Carrying out the sketch of the second modification is much more involved and is a corollary of a recent work by Feige and Shamir [28]. Assuming the existence of a bit commitment scheme (the same assumption as the one made in this paper), they show that any language in NP has a bounded-round (computational) zero-knowledge argument. Using a stronger assumption, Brassard et al. independently showed that any language in NP has a bounded-round *perfect* zero-knowledge argument [18]. The reader should note that the three results mentioned above are incomparable: Goldreich and Kahn use a specific intractability assumption and get a (computational) zero-knowledge proof, Feige and Shamir use a general complexity assumption and get a (computational) zero-knowledge argument, while Brassard et al. use a specific intractability assumption (incomparable to the one of Goldreich and Kahn) and get a perfect zero-knowledge argument.

Remark 15. Protocol 4 is zero-knowledge assuming that the encryption function used is *nonuniformly* secure. We do not know whether Protocol 4

remains zero-knowledge if the encryption function is "only" (uniformly) secure (i.e., cannot be "broken" by polynomial-time *algorithms*). A difficulty encountered when trying to prove the statement is that the existence of a sequence of graphs for which the simulator fails does not yield a contradiction to the security of the encryption function, since there may be no efficient way to generate these "problematic" graphs. However, a weaker statement can be proven. An interactive proof system will be called "zero-knowledge on sampleable distributions" if it is infeasible to find an instance (i.e., a triple (x, y, z), where x is the common input, y is a private input for the prover and z is a private input for the verifier) on which the simulator fails. For details, see Goldreich [39].

Remark 16. Allowing the simulator to be expected polynomial-time (rather than strictly probabilistic polynomial-time) is not essential to the zero-knowledge property of Protocol 4, provided that the verifier is also restricted to be strictly probabilistic polynomial-time. An easy modification of the simulation makes it strictly polynomial-time without losing much in the quality of the conversations produced. In the modified simulation, each round is repeated at most m^2 times and the simulator stops in the (highly unlikely) case that all m^2 tries where unsuccessful. Thus, using "strictly polynomial time" instead of "expected polynomial time" in the definition of zero-knowledge, we also get that Protocol 4 is zero-knowledge.

3.2 Zero-Knowledge Proof Systems for All NP. The following theorem asserts that every language $L \in \mathrm{NP}$ has a zero-knowledge proof system. We know of three alternative ways of proving the theorem. The first two ways consist of incorporating the standard reductions into a protocol for 3-colorability (either an arbitrary protocol or Protocol 4). A subtle difficulty arises from the fact that the cheating verifier has $x \in L$ and not only the graph into which x is mapped. This might assist him when trying to extract "knowledge" from the proof that the corresponding graph is 3-colorable. We show that this cannot be the case by using additional facts. The first alternative is to use the fact that the reduction is invertible; while the second alternative is to use the fact that Protocol 4 has been proven zero-knowledge using a black-box simulation (and thus is "auxiliary input zero-knowledge" [44, 60]). The third way to prove the theorem consists of presenting a zero-knowledge proof system for L directly, that is, using the nondeterministic circuit recognizing L (M. Blum, private communication). Here, we use the first alternative.

Theorem 5. *If there exists a nonuniformly secure encryption function, then every language in NP has a zero-knowledge interactive proof system.*

Proof. (For NP-Completeness terminology and results consult [34].) Let $L \in \mathrm{NP}$, and t be the polynomial-time computable and invertible reduction of L to 3-Colorability ($G3C$). Namely, t is the composition of the standard reduction of L to 3SAT (obtained by Cook's proof) and the standard reduction of 3SAT to $G3C$ (presented in [34]). Recall that $x \in L$ iff $t(x)$ is 3-colorable. A zero-knowledge interactive proof system for L proceeds as follows:

On common input x, each party computes $G \leftarrow t(x)$. The prover uses an (arbitrary) zero-knowledge interactive proof system to prove that G is 3-colorable. The verifier acts according to the result of this subprotocol.

Clearly, the above protocol constitutes an interactive proof system for L. To see that the protocol is indeed zero-knowledge, one should note that t is polynomial-time invertible (i.e., there exists a polynomial-time algorithm t^{-1} such that $t^{-1}(t(x)) = x$). Details follow:

Let V^* be an ITM that interacts with P, the prover specified for L. Note that P applies algorithm t to the input x and initiates P_{G3C}, the prover specified for $G3C$, on $t(x)$. We consider the ITM V^{**} which interacts with P_{G3C} on common input $t(x)$ and auxiliary input x (i.e., $\langle P_{G3C}, V^{**}(x)\rangle(t(x)) = \langle P, V^*\rangle(x)$). Note that V^{**}, which has an auxiliary input, is not of the form allowing the application of the fact that P_{G3C} is a zero-knowledge prover. Instead, we construct an ITM V^{***} that on input $t(x)$, first computes x and then applies V^{**}. (Here, we use the fact that t is polynomial-time invertible.) Now, by the fact that P_{G3C} is a zero-knowledge prover, it follows that there exists an $M_{V^{***}}$ such that $\{\langle P_{G3C}, V^{***}\rangle(t(x))\}_{x\in L}$ and $\{M_{V^{***}}(t(x))\}_{x\in L}$ are polynomial indistinguishable. Let $M_{V^*}(x) = M_{V^{***}}(t(x))$. Since $\langle P, V^*\rangle(x)$ equal $\langle P_{G3C}, V^{***}\rangle(t(x))$, the theorem follows. □

Theorem 6 adapts Theorem 5 to a cryptographic scenario in which all players are bounded to efficient computation. What is needed is to notice that the standard reductions also provide (implicitly) *efficient* transformation between the solutions to the instances.

THEOREM 6. *If there exists a nonuniformly secure encryption function, then every language in NP has a zero-knowledge interactive proof system in which the prover is a probabilistic polynomial-time machine that gets an NP proof as an auxiliary input. (An NP proof of "$x \in L$" is a sequence of nondeterministic choices leading a fixed NP machine to accept x. Formally, let P_L: $\{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be a polynomial-time computable predicate such that $x \in L$ iff $\exists y$ such that $P_L(x, y) = 1$. Then, y is called an NP proof of "$x \in L$".)*

PROOF. Let t be as in the proof of Theorem 5. Let t' be a polynomial-time computable function transforming the sequence of nondeterministic choices leading to acceptance of x into a proper 3-coloring of the graph $t(x)$ (i.e., if $P_L(x, y) = 1$, then $t'(y)$ is a 3-coloring of $t(x)$). Note that the fact that such a function is polynomial-time computable is guaranteed by neither Karp's nor Cook's definition of NP-completeness.[12] Nevertheless, the standard reduction of L to $G3C$, has this property. Recall that the prover in Protocol 4 may be a probabilistic polynomial-time machine that gets a 3-coloring as an auxiliary input. It follows that the prover in the protocol presented in the proof of Theorem 5 may be a probabilistic polynomial-time machine that receives as input $x \in L$ and y such that $P_L(x, y) = 1$. (The prover computes $G \leftarrow t(x)$ and $\phi \leftarrow t'(y)$, and participates in Protocol 4 using the 3-coloring ϕ of G.) The theorem follows. □

Remark 17. Another useful strengthening of the results of Theorems 5 and 6, is to prove that the interactive proof systems suggested above are in fact "auxiliary input zero-knowledge." Intuitively, this means that whatever the verifier can efficiently compute after interacting with the prover on input $x \in L$ and *having additional information* z, can be efficiently computable from x

[12] Interestingly, polynomial-time computable transformation of witnesses is guaranteed by Levin's definition of NP-completeness [56].

and z (without interacting with anybody). Here the auxiliary input is to the verifier who, even with it in his possession, cannot extract "knowledge" from the prover. It is easy to see that Protocol 4 as well as the protocol presented in the proof of Theorem 5 are in fact auxiliary-input zero-knowledge. For further details see [44] and [60]. The reason that this extension is important to cryptographic applications is that in typical applications, zero-knowledge proof systems are used as subprotocols inside another protocol. The party playing the role of the verifier may have extra information, obtained in previous stages of the protocol, which he might use in the (zero-knowledge) subprotocol in order to try to extract "knowledge" from his counterpart (who plays the prover).

Remark 18. Another issue of importance for practical applications is the efficiency of a zero-knowledge proof system, and in particular the efficiency of a zero-knowledge proof system for a language L as function of the nondeterministic Turing machine complexity of L. There are two standard efficiency measures that may be considered:

(1) The *computational complexity* of the proof system (i.e., number of steps taken by either or both parties).
(2) The *communication complexity* of the proof system. Here one may consider the number of interactions, and/or the total number of bits exchanged.[13] (For example, in Protocol 4 the number of interactions is $O(m^2)$ and the number of bits exchanged is $O(n^2 \cdot m^2)$.)

A nonstandard measure for the efficiency of a zero-knowledge proof system is its *tightness*. Intuitively, tightness is the ratio between the time it takes the simulator to simulate an interaction with the prover and the time the interaction really takes. Formally, the *knowledge-tightness* (*tightness*) of a zero-knowledge proof system is a function $t\colon N \to N$ (from integers to integers) satisfying the following: For every polynomial-time ITMs V^*, there exists a machine M_{V^*} such that $\{M_{V^*}(x)\}_{x\in L}$ is polynomially indistinguishable from $\{\langle P, V^*\rangle(x)\}_{x\in L}$, and

$$\frac{time(M_{V^*}(x))}{time(V^*(x))} < t(|x|),$$

where $time(A(x))$ is the expected number of steps taken by the machine (resp., ITM) A on input x. The definition of zero-knowledge only guarantees that the knowledge-tightness does not have to grow faster than a polynomial. However, the definition does not guarantee that the knowledge-tightness can be bounded above by a *particular* polynomial. It is easy to see that the knowledge-tightness of Protocols 2 and 3 is a constant while the tightness of Protocol 4 is m (i.e., the number of edges). We believe that the knowledge-tightness of a protocol is the most important efficiency measure to be considered, and that it is very desirable to have it be a constant. Furthermore, using the notion of knowledge-tightness one can introduce more refined notions of zero-knowledge and in particular constant-tightness zero-knowledge. Such refined notions may be applied in a nontrivial manner also to languages in P.

[13] A third measure, the importance of which has been realized only recently (see [55]), is the number of "locked boxes" to be used in a physical implementation of the system.

The protocols obtained in Theorems 5 and 6 are not the most efficient possible. Protocols with constant knowledge-tightness and a number of iterations which is (merely) super-logarithmic exist for all languages in NP (assuming, of course, the existence of secure encryption). Suggestions are due to J. Benaloh, M. Blum, D. Chaum, R. Impagliazzo, M. Rabin, A. Shamir and possibly others. The most efficient suggestion (concurrently in all measures) is a protocol that uses the circuit value problem. For more details, see [37].

3.3. Applications—An Example. Owing to its generality, Theorem 6 has a dramatic effect on the design of cryptographic protocols. Let us demonstrate this point by using Theorem 6 to present a simple solution to a problem that until recently was considered very complex: *verifiable secret sharing*. The more general implications of Theorem 6 are investigated in [42] and [43].

The notion of a verifiable secret sharing was presented by Chor et al. [21], and constitutes a powerful tool for multiparty protocol design. Loosely speaking, a *verifiable secret sharing* is an $n + 1$-party protocol through which a *sender* can distribute, to then *receivers*, *pieces* of a secret s recognizable through an a priori known "encryption" $f(s)$. The n pieces should satisfy the following three conditions (with respect to $1 \leq l < u \leq n$):

(1) It is infeasible to obtain any knowledge about the secret from any l pieces;
(2) Given any u pieces, the entire secret can be easily computed;
(3) Given a piece, it is easy to verify that it belongs to a set satisfying condition (2).

The notion of a verifiable secret sharing differs from Shamir's secret sharing [65] (invented independently by Blakely [13]), in that the secret is recognizable and that *the pieces should be verifiable as authentic* (i.e., condition (3)).

Following the first implementation presented in [21], improvements in efficiency and "tolerance" appeared in [5] and (P. Feldman and S. Micali, private communication). These solutions are conceptually complicated, and rely on specific properties of particular encryption functions.

Assuming the existence of *arbitrary* one-way permutations, we present a conceptually simple solution allowing $u = l + 1 \leq n$. Our scheme combines Theorem 6 with Shamir's (nonverifiable) secret sharing [65]. Let p be a prime and $p \geq n$. To share a secret $s \in GF(p)$ recognizable through $r = f(s)$, the sender proceeds as follows: First, the sender chooses at random an l-degree polynomial over $GF(p)$ with free term s, and evaluates it in n fixed nonzero points (these are the pieces in Shamir's scheme). Next, the sender encrypts the ith piece using the public encryption algorithm of the ith receiver, and sends all encrypted secrets to all receivers. Finally, the sender provides each receiver with a zero-knowledge proof that the encrypted pieces correspond to the evaluation of a single polynomial over $GF(p)$, and that applying f to the free term of this polynomial yields s. This statement is an NP-statement and hence can be proven in zero-knowledge.

Recently, Feldman found a more efficient implementation of verifiable secret sharing based on the intractability of factoring [29].

4. *Conclusions*

The primary motivation for the concept of zero-knowledge proof systems has been their potential use in the design of cryptographic protocols. Early exam-

ples of such use can be found in [23], [30], and [47]. The general results in this paper allowed the presentation of automatic generators of two-party and multiparty cryptographic protocols (see [69] and [43], respectively). Further improvements are reported in [33], [45], and [53].

An elegant model for the investigation of multiparty cryptographic protocols was suggested in [12]. This model consists of processors connected in pairs via *private channels*. The bad processors have infinite computing resources (and so using computationally hard problems is useless). Hence, computational complexity restrictions and assumptions are substituted by assumptions about the communication model. Work in this model is inspired by our results and the results in [43] but does not use them explicitly. In particular, an automatic generator of protocols for this model, tolerating up to a $< \frac{1}{3}$ fraction of malicious processors, has been presented in [12] and [20]. Augmenting the model by a broadcast channel, tolerance can be improved to $< \frac{1}{2}$ fraction [62]. (The augmentation is necessary, as there are tasks that cannot be performed if a third of the processors are malicious (e.g., Byzantine Agreement).) Beyond the $\frac{1}{2}$ bound, only functions of special type (e.g., in the Boolean case only, the exclusive-OR of locally computed functions) can be privately computed [22].

4.1. A "POSITIVE" USE OF NP-COMPLETENESS. So far NP-completeness results have mostly had a "negative" utility: It was (and is) the most practical way to give evidence to the intractability of a problem. Here, we want to point out a "positive" use of NP-completeness: Its primary role in deriving the general results of Theorems 5 and 6 (i.e., zero-knowledge proof systems for every NP-statement) from Proposition 4 (i.e., a zero-knowledge proof system for a particular NP-complete problem).

ACKNOWLEDGMENTS. We wish to thank Baruch Awerbuch, Manuel Blum, Benny Chor, Shimon Even, Mike Fischer, Shafi Goldwasser, Dick Karp, Leonid Levin, Albert Meyer, Yoram Moses, Michael Rabin, Charlie Rackoff, Ron Rivest, and Mike Sipser for helpful discussions concerning this work.

We are grateful to Josh (Cohen) Benaloh for suggesting the simplification in Protocol 3. Special thanks to Gilles Brassard, Ariel Kahn, Hugo Krawczyk, Eyal Kushilevitz, Yair Oren, and the anonymous referees for their remarks on earlier versions of this manuscript.

REFERENCES

1. AHO, A. V., HOPCRAFT, J. E., AND ULLMAN, J. D. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, Mass., 1974.
2. AIELLO, W., GOLDWASSER, S., AND HASTAD, J. On the power of interaction. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1986, pp. 368–379.
3. AIELLO, W., AND HASTAD, J. Perfect zero-knowledge languages can be recognized in two rounds. In *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1987, pp. 439–448.
4. ALEXI, W., CHOR, B., GOLDREICH, O., AND SCHNORR, C. P. RSA and Rabin functions: Certain parts are as hard as the whole *SIAM J. Comput. 17*, 2 (1988), 194–209. (Extended abstract in *Proceedings of the 25th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1984.)
5. ALON, N., GALIL, Z., AND YUNG, M. A fully polynomial simultaneous broadcast in the presence of faults. Unpublished manuscript, 1985.

6. BABAI, L. Trading group theory for randomness. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (Providence, R.I., May 6–8). ACM, New York, 1985, pp. 421–429.
7. BABAI, L., KANTOR, W. M., AND LUKS, E. M. Computational complexity and classification of finite simple groups. In *Proceedings of the 24th Annual IEEE Foundations of Computer Science*. IEEE, New York, 1983, pp. 162–171.
8. BABAI, L., AND MORAN, S. Arthur–Merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci. 36*, 2 (1988), 254–276.
9. BELLARE, M., MICALI, S., AND OSTROVSKY, R. Perfect zero-knowledge in constant rounds. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing* (Baltimore, Md., May 12–14). ACM, New York, 1990, pp. 482–493.
10. BENALOH (COHEN), J. D. Cryptographic capsules: A disjunctive primitive for interactive protocols. In A. M. Odlyzko, ed., *Proceedings of Advances in Cryptology—Crypto86*. Lecture Notes in Computer Science, vol. 263. Springer-Verlag, New York, 1987, pp. 213–222.
11. BEN-OR, M., GOLDREICH, O., GOLDWASSER, S., HASTAD, J., KILLIAN, J., MICALI, S., AND ROGAWAY, P. Everything provable is provable in zero-knowledge. In *Proceedings of Advances in Cryptology—Crypto88*. Lecture Notes in Computer Science, vol. 403. Springer-Verlag, New York, 1990, pp. 37–56.
12. BEN-OR, M., GOLDWASSER, S., AND WIGDERSON, A. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (Chicago, Ill., May 2–4). ACM, New York, 1988, pp. 1–10.
13. BLAKELY, G. R. Safeguarding cryptographic keys. In *Proceedings of National Computer Conference*, vol. 48. AFIPS Press, 1979, pp. 313–317.
14. BOPPANA, R., HASTAD, J., AND ZACHOS, S. Does co-NP have short interactive proofs? *Inf. Proc. Lett. 25* (May 1987), 127–132.
15. BRASSARD, G., CHAUM, D., AND CRÉPEAU, C. Minimum disclosure proofs of knowledge. *J. Comput. Syst. Sci. 37*, 2 (1988), 156–189.
16. BRASSARD, G., AND CRÉPEAU, C. Zero-knowledge simulation of Boolean circuits. In A. M. Odlyzko, ed., *Proceedings of Advances in Cryptology—Crypto86*. Lecture Notes in Computer Science, vol. 263. Springer-Verlag, New York, 1987, pp. 223–233.
17. BRASSARD, G., AND CRÉPEAU, C. Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for SAT and beyond. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1986, pp. 188–195.
18. BRASSARD, G., CRÉPEAU, C., AND YUNG, M. Everything in NP can be argued in *perfect* zero-knowledge in a *constant* number of rounds. In *Proceedings of the 16th Annual International Colloquium on Automata Languages and Programming*. Lecture Notes in Computer Science, vol. 435. Springer-Verlag, New York, 1989, pp. 123–136.
19. CHAUM, D. Demonstrating that a public predicate can be satisfied without revealing any information about how. In A. M. Odlyzko, ed., *Proceedings of Advances in Cryptology—Crypto86*. Lecture Notes in Computer Science, vol. 263. Springer-Verlag, New York, 1987, pp. 195–199.
20. CHAUM, D., CRÉPEAU, C., AND DAMGARD, I. Multiparty unconditionally secure protocols. In *Proceedings of the 20th Annual ACM Symposium on Theory of Computing* (Chicago, Ill., May 2–4). ACM, New York, 1988, pp. 11–19.
21. CHOR, B., GOLDWASSER, S., MICALI, S., AND AWERBUCH, B. Verifiable secret sharing and achieving simultaneity in the presence of faults. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1985, pp. 383–395.
22. CHOR, B., AND KUSHILEVITZ, E. A zero–one law for Boolean privacy. In *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing* (Seattle, Wash., May 15–17). ACM, New York, 1989, pp. 62–72.
23. COHEN, J. D., AND FISCHER, M. J. A robust and verifiable cryptographically secure election scheme. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1985, pp. 372–382.
24. COOK, S. A. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on the Theory of Computing* (Shaker Heights, Ohio, May 3–5). ACM, New York, 1971, pp. 151–158.
25. DIFFIE, W., AND HELLMAN, M. E. New directions in cryptography. *IEEE Trans. Inf. Theory, IT-22*, 6 (Nov. 1976), 644–654.

26. EVEN, S., GOLDREICH, O., AND LEMPEL, A. A randomized protocol for signing contracts. *Commun. ACM 28*, 6 (June 1985), 637–647.
27. FEIGE, U., FIAT, A., AND SHAMIR, A. Zero-knowledge proofs of identity. *J. Crypto. 1*, 2 (1988), 77–94.
28. FEIGE, U., AND SHAMIR, A. Zero-knowledge proofs of knowledge in two rounds. In *Proceedings of Advances in Cryptology—Crypto89*. Lecture Notes in Computer Science, vol. 435. Springer-Verlag, New York, 1990, pp. 526–544.
29. FELDMAN, P. A practical scheme for verifiable secret sharing. In *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1987, pp. 427–438.
30. FISCHER, M., MICALI, S., RACKOFF, C., AND WITTENBERG, D. K. An oblivious transfer protocol equivalent to factoring. Unpublished manuscript, 1986.
31. FORTNOW, L. The complexity of perfect zero-knowledge. In *Proceedings of the 19th Annual ACM Symposium on the Theory of Computing* (New York, N.Y., May 25–27). ACM. New York, 1987, pp. 204–209.
32. GALIL, Z., HABER, S., AND YUNG, M. A private interactive test of a Boolean predicate and minimum-knowledge public-key cryptosystems. In *Proceedings of the 26th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, pp. 360–371.
33. GALIL, Z., HABER, S., AND YUNG, M. Cryptographic computation: Secure fault-tolerant protocols and the public-key model. In C. Pomerance, ed., *Proceedings of Advances in Cryptology—Crypto87*. Lecture Notes in Computer Science, vol. 293. Springer-Verlag, New York, 1987, pp. 135–155.
34. GAREY, M. R., AND JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.
35. GAREY, M. R., JOHNSON, D. S., AND STOCKMEYER, L. Some simplified NP-complete graph problems. *Theoret. Comput. Sci. 1* (1976), 237–267.
36. GOLDREICH, O. A zero-knowledge proof that a two-prime moduli is not a Blum integer. Unpublished manuscript, 1985.
37. GOLDREICH, O. Zero-knowledge and the design of secure protocols (an exposition). Tech. Rep. TR-480. Comput. Sci. Dept., Technion, Haifa, Israel, 1987.
38. GOLDREICH, O. Towards a theory of average case complexity (a survey). Tech. Rep. TR-531. Comput. Sci. Dept., Technion, Haifa, Israel, 1988.
39. GOLDREICH, O. A uniform-complexity treatment of encryption and zero-knowledge. Tech. Rep. TR-568. Comput. Sci. Dept., Technion, Haifa, Israel, 1989.
40. GOLDREICH, O., AND KRAWCZYK, H. On the composition of zero-knowledge proof systems. In *Proceedings of the 17th Annual International Calloquium on Automata Languages and Programming*. Lecture Notes in Computer Science, vol. 443. Springer-Verlag, New York, 1990, pp. 268–282.
41. GOLDREICH, O., MANSOUR, Y., AND SIPSER, M. Interactive proof systems: Provers that never fail and random selection. In *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1987, pp. 449–461.
42. GOLDREICH, O., MICALI, S., AND WIGDERSON, A. Proofs that yield nothing but their validity and a methodology of cryptographic protocol design. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1986, pp. 174–187.
43. GOLDREICH, O., MICALI, S., AND WIGDERSON, A. How to play ANY mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing* (New York, N.Y., May 25–27). ACM, New York, 1987, pp. 218–229.
44. GOLDREICH, O , AND OREN, Y. Definitions and properties of zero-knowledge proof systems. Tech. Rep. TR-610. Comput. Sci. Dept., Technion, Haifa, Israel, 1990.
45. GOLDREICH, O., AND VAINISH, R. How to solve any protocol problem—An efficiency improvement. In C. Pomerance, ed., *Proceedings of Advances in Cryptology—Crypto87*. Lecture Notes in Computer Science, vol. 293. Springer-Verlag, New York, 1987, pp. 73–86.
46. GOLDWASSER, S., AND MICALI, S. Probabilistic encryption. *J. Comput. Syst. Sci. 28*, 2 (1984), 270–299.
47. GOLDWASSER, S., MICALI, S., AND RACKOFF, C. The knowledge complexity of interactive proof systems. *SIAM J. Comput. 18*, 1 (1989), 186–208.
48. GOLDWASSER, S., MICALI, S., AND RIVEST, R. L. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput. 17*, 2 (1988), 281–308.

49. GOLDWASSER, S., AND SIPSER, M. Private coins versus public coins in interactive proof systems. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing* (Berkeley, Calif., May 28–30). ACM, New York, 1986, pp. 59–68.
50. GUREVICH, Y. Average case completeness. Tech. Rep. CRL-TR-03-88, Comput. Res. Lab., Univ. Michigan, Ann Arbor, Mich., 1988.
51. HÅSTAD, J. Pseudo-random generators under uniform assumptions. In *Proceedings of the 22nd Annual ACM Symposium on Theory of Computing* (Baltimore, Md., May 12–14). ACM, New York, 1990, pp. 395–404.
52. IMPAGLIAZZO, R., LEVIN, L. A., AND LUBY, M. Pseudorandom generation from one-way functions. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing* (Seattle, Wash., May 15–17). ACM, New York, 1989, pp. 12–24.
53. IMPAGLIAZZO, R., AND YUNG, M. Direct minimum-knowledge computations. In C. Pomerance, ed., *Proceedings of Advances in Cryptology—Crypto87*. Lecture Notes in Computer Science, vol. 293. Springer-Verlag, New York, 1987, pp. 40–51.
54. KARP, R. M. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, eds. *Complexity of Computer Computations*. Plenum Press, New York, 1972, pp. 85–103.
55. KILIAN, J., MICALI, S., AND OSTROVSKY, R. Minimum resource zero-knowledge proofs. In *Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1989, pp. 474–479.
56. LEVIN, L. A. Universal search problems. *Prob. Pere. Inf. 9* (1973), 115–116. Translated in *Prob. Inf. Trans. 9* (1973), 265–266.
57. LEVIN, L. A. Average case complete problems. *SIAM J. Comput. 15* (1986), 285–286.
58. NAOR, M. Bit commitment using pseudorandomness. In *Proceedings of Advances in Cryptology—Crypto89*. Lecture Notes in Computer Science, vol. 435. Springer-Verlag, New York, 1990, pp. 128–137.
59. NISSAN, N., AND WIGDERSON, A. Hardness vs. randomness. In *Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1988, pp. 2–11.
60. OREN, Y. On the cunning power of cheating verifiers: Some observations about zero-knowledge proofs. In *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1987, pp. 462–471.
61. RABIN, M. O. Digitalized signatures and public-key functions as intractable as factorization. Tech. Rep. MIT/LCS/TR-212. MIT, Cambridge, Mass., 1979.
62. RABIN, T., AND BEN-OR, M. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing* (Seattle, Wash., May 15–17). ACM, New York, 1989, pp. 73–85.
63. RIVEST, R. L., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM 21*, 2 (Feb. 1978), 120–126.
64. SCHOENING, U. Graph isomorphism is in the low hierarchy. In *Proceedings of the Symposium on Theoretical Aspects of Computer Science (STACS 87)*. Lecture Notes in Computer Science, vol. 247. Springer-Verlag, New York, 1987, pp. 114–124.
65. SHAMIR, A. How to share a secret. *Commun. ACM 22*, 11 (Nov. 1979), 612–613.
66. STOCKMEYER, L. J. The polynomial-time hierarchy. *Theoret. Comput. Sci. 3* (1977), 1–22.
67. TOMPA, M., AND WOLL, H. Random self-reducibility and zero-knowledge interactive proofs of possession of information. In *Proceedings of the 28th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1987, pp. 472–482.
68. YAO, A. C. Theory and applications of trapdoor functions. In *Proceedings of the 23rd Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1982, pp. 80–91.
69. YAO, A. C. How to generate and exchange secrets. In *Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science*. IEEE, New York, 1986, pp. 162–167.

RECEIVED APRIL 1988; REVISED MARCH 1989, SEPTEMBER 1989, AND MARCH 1990; ACCEPTED MAY 1990