PROOFS THAT YIELD NOTHING BUT THEIR VALIDITY
OR
ALL LANGUAGES IN NP HAVE ZERO-KNOWLEDGE PROOFS

by

O. Goldreich, S. Micali and A. Wigderson

Technical Report #498
February 1988
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 polynomially computable Boolean predicate associated to the language $L$ such that $P_L(x, y) = 0$ for all $y$ if $x$ is not in $L$. 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 polynomial-time deterministic 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 [GMR, B]. The prover has some computational 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). In other words, we will implicitly assume that there exist interesting languages (say in $PSPACE$) which are not in $BPP$, and be interested in proof systems for such languages.

A fundamental measure proposed by Goldwasser, Micali and Rackoff [GMR] is that of the amount of knowledge released during an interactive proof. Informally, a proof system was called zero-knowledge if whatever the verifier could generate in probabilistic polynomial-time after "seen" 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. In other words, 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 [FMRW, GMR, CF]. However, in order to demonstrate the generality of this tool (and to utilize its full potential) one should have come with general results concerning the existence of zero-knowledge proof systems. Until now, no such general results were obtained.

In this paper, we present general results concerning zero-knowledge proof systems. In particular, we show how to give zero-knowledge proofs to 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 [GMW1, GMW2], 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 behaviour).
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, when interacting with the prover. If $x \notin L$ then no matter what the prover does, he cannot fool the verifier (into believing that "$x$ is in $L""), except for with very low probability. 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, we define the notion of an interactive pair of Turing machines, which captures the intuitive notion of a two-party protocol.

Definition 1 (Interactive Turing machines [Bl2]): 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 which appears on the input tape is called the input. (The contents of the random tape can be thought of as the outcomes of unbiased coin tosses.) The string which 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.)

An interactive Turing machine $M$ is polynomial-time if there exists a polynomial $Q$ such that on input $x$ machine $M$ performs at most $Q(|x|)$ steps, where $|x|$ denotes the length of the string $x$. (In one step an ITM, depending upon the symbols scanned by the tapes' heads and its control state, may change its state, print a symbol on one of the scanned tape cells, and move one of its tape-heads by one cell.)

An interactive pair of Turing machines is a pair of ITMs which share their communication tapes so that the read-only (communication) tape of the first machine is 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 start state. The string written on the communication tape during a single non-alternating 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 $\alpha_1$ through $\alpha_t$.

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 the cartesian product of all possible coin tosses of each machine, taken with uniform probability distribution.
Definition 2 (Interactive Proof [GMR]): An interactive proof for a language $L$ is a pair of ITMs, $<P, V>$, such that $V$ is polynomial-time and the following two conditions hold:

1) **Completeness condition:** For every constant $c > 0$, and all sufficiently long $x \in L$,
   \[
   \Pr\left[ (P(x), V(x)) = 1 \right] \geq 1 - 1/|x|^c.
   \]

2) **Soundness condition:** For every constant $c > 0$, every interactive Turing machine $P^*$, all sufficiently long $x \not\in L$, and every $y \in \{0,1\}^*$,
   \[
   \Pr\left[ (P^*(y), V(x)) = 0 \right] \geq 1 - 1/|x|^c.
   \]

Denote by IP the class of languages having interactive proofs.

**Remark 1:** An important example of an interactive proof system 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 for the language $L$ does not imply the existence of an interactive proof for the complement of $L$.

**Remark 3:** Definition 2 originates from Goldwasser, Micali and Rackoff [GMR]. A different definition due to Babai [B], restricts the verifier to generate random strings, send them to the prover, and evaluate a deterministic polynomial-time predicate at the end of the interaction. Babai's formulation is known as Arthur-Merlin games. Demonstrating the existence of proof systems is easier when allowing the verifier to flip private coins (i.e. [GMR] model), 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 [GS]. Namely, let $L$ be a language having an interactive proof in which up to $Q(1 \cdot |x|)$ messages are exchanged on input $x \in L$. Then, $L$ has an Arthur-Merlin interactive-proof in which up to $Q(1 \cdot |x|)$ messages are exchanged on input $x \in L$.

**Remark 4:** The ability to toss coins is crucial to the non-triviality of the notion of an interactive proof system. If the verifier is deterministic then interactive proof systems coincide with NP.

**Remark 5:** Without loss of generality, we assume that the last message sent during an interactive proof is sent by the prover. (A last message sent by the verifier has absolutely no effect.)

Babai [B] showed that for every polynomial $Q$, \( AM(Q+1) = AM(Q) \), where $AM(Q)$ denotes the class of languages recognized by an Arthur-Merlin game of $Q(1 \cdot |1|)$ steps. This means that the finite level Arthur-Merlin hierarchy (as well as the finite level IP hierarchy) collapses (i.e. for every fixed $k \geq 2$, $AM(k) = AM(2)$). Note that this does not imply the collapse of the unbounded level hierarchy! (For more details see [AGH].) The bounded level interactive proofs hierarchy is related to the polynomial-time hierarchy by Babai's proof that $AM(2) \subseteq \Pi^P_2$ (the second level) and that $AM(2) \subseteq NP^B$ for almost all oracles $B$.
1.2. What is a Zero-Knowledge Proof

Intuitively, a zero-knowledge proof is a proof which yields nothing but its validity. This means that for all practical purposes, "whatever" can be efficiently computed after interacting with a zero-knowledge prover, can be efficiently computed when just believing that the assertion he claims is indeed valid. (In "whatever" we mean not only the computation of functions but also the generation of probability distributions.) Thus, zero-knowledge is a property of the predetermined prover. It is the robustness of the prover 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 [GMR]): Let \( <P, V> \) be an interactive proof system for a language \( L \), and \( V^* \) be an arbitrary ITM. Denote by \( V^*(x) \) the probability distribution of the output of \( V^* \) when interacting with \( P \) (the prover) on common input \( x \in L \). We say that the proof system \( <P, V> \) is \( \text{zero-knowledge} \) (for \( L \)) if for all polynomial-time ITM \( V^* \), there exists a probabilistic polynomial-time machine \( M_{V^*} \) such that the probability distributions \( \{M_{V^*}(x)\}_{x \in L} \) and \( \{V^*(x)\}_{x \in L} \) are polynomially-indistinguishable (see Definition 4 below).

The machine \( M_{V^*} \) is called the simulator of \( V^* \). Being able to simulate \( V^* \) means that the output of \( V^* \) can be produced also without interacting with the prover (namely by \( M_{V^*} \)). The universal quantifier states that this is the case with respect to any interaction with the prover. In fact, \( M_{V^*} \) may not be able to exactly simulate \( V^* \)'s interaction with the prover, however, it can produce something which is indistinguishable. For the sake of selfcontainment, we recall the definition of polynomial-indistinguishable ensembles [GM, Y1].

Definition 4 (Polynomial Indistinguishability [GM, Y1]): 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 \( p^A_i(x) \) denote the probability that \( A \) outputs 1 on input \( x \) and an element chosen according to the probability distribution \( \Pi_i(x) \). Namely,

\[
p^A_i(x) = \sum_\alpha \Pr[A(x,\alpha) = 1] \cdot \Pr[\Pi_i(x) = \alpha].
\]

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

\[
|p^A_1(x) - p^A_2(x)| \leq 1 \cdot \text{poly}^{-c}.
\]

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

February 23, 1988
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 $<P,V>$ be an interactive proof system for a language $L$, and $V^*$ be an arbitrary ITM. Denote by $<P,V^*>(x)$ 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 $<P,V>$ is zero-knowledge (for $L$) if for all polynomial-time ITM $V^*$, there exists a probabilistic polynomial-time machine $M_{V^*}$ such that the probability distributions $\{M_{V^*}(x)\}_{x \in L}$ and $\{<P,V^*>(x)\}_{x \in L}$ 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. $\forall V^* \exists T^* \text{ s.t. } T^* (<P,V^*>(x)) = V^*(x)$). On the other hand, every $V^*$ can be easily modified to output the contents of its read-only tapes (i.e. $\forall V^* \exists V^{**} \text{ s.t. } V^{**}(x) = <P,V^*>(x)$). However, it is not clear that ability to simulate the output of a specific $V^*$ implies ability to simulate the contents of $V^*$'s tapes. In particular, the reader should consider the meaning of these two requirements with respect to the prescribed verifier, $V$.

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

$$<P,V^*>(x) = M_{V^*}(x).$$

An interactive proof is almost-perfect zero-knowledge if for every $x \in L$ the distributions $<P,V^*>(x)$ and $M_{V^*}(x)$ are "statistically close", i.e. the statistical difference (defined below) between $<P,V^*>(x)$ and $M_{V^*}(x)$ is smaller than $\epsilon x^{-}\epsilon$, for all $\epsilon > 0$ 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 of 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 $A$ (which was required above to be polynomial-time).

**Remark 8:** It is important to stress that in our model of interactive proofs the internal computations of the parties are not assumed to be simultaneous. Thus, one machine can not 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 of the NP-type then $L$ in Random polynomial-time. (A proof can be found in [GO, Ore].) Thus, extending $NP$ to $IP$ is essential to the non-triviality of the notion of zero-knowledge.
Several Number Theoretic languages, *not known to be in BPP*, have been previously shown to have zero-knowledge proof systems. The first language for which such a proof system has been demonstrated is Quadratic Non-Residuosity [GMR]. Other zero-knowledge proof systems were presented in [GMR], [GHY], [CF] and [G1]. All these languages are known to lie in \( NP \cap \text{Co-NP} \).

### 1.3. Our Results

Under the assumption that secure encryption functions exist or by using "physical means for hiding information", we show that all languages in \( NP \) have zero-knowledge proofs. 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 proofs exist "independently of cryptography and number theory". Using no unproved assumptions, we show that both graph isomorphism and graph non-isomorphism possess zero-knowledge interactive proofs.

### 1.4. Related Results

Using the intractability assumption of quadratic residuosity, Brassard and Crepeau have discovered subsequently zero-knowledge proof systems to all languages in \( NP \) [BC1]. These proof systems heavily rely on particular properties of quadratic residues and do not seem to extend to arbitrary encryption functions.

Recently, Brassard and Crepeau showed that if factoring is intractable then every \( NP \) language has a perfect zero-knowledge *pseudo-proof* [BC2]. By a *pseudo-proof* we mean that the verifier is convinced only if he believes both that the prover is a polynomial-time machine (with some auxiliary input which is fixed before the protocol starts), and that the intractability assumption does hold. This should be confronted with the soundness condition in [GMR] and in this paper which does not depend on intractability assumptions nor restricts the computational power of the prover.

Independently, Chaum [Cha] discovered a protocol which is very similar to the one in [BC2]. Chaum also proposed an interesting application of such "perfect zero-knowledge pseudo-proofs". His application is to a setting in which the verifier may have infinite computing power while the prover is restricted to polynomial-time computations (see also [CEGP]). In such a setting it makes no sense to have the prover demonstrate 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 formulae, the prover wishes 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 [GMR], and recently formalized in [FFS, TW].

February 23, 1988
1.5. Some Subsequent Results Concerning Interactive Proof Systems and Zero-Knowledge

Fortnow has recently shown that if a language $L$ has a perfect (or even almost-perfect) zero-knowledge proof system then its complement (i.e. $\overline{L}$) has a two-step interactive proof [For]. It follows that, unless all $co-NP$ has interactive proofs, NP-Complete languages can not have perfect (or almost-perfect) zero-knowledge proofs. In fact, Fortnow gives a transformation of the zero-knowledge proof of $L$ into an interactive proof for $\overline{L}$. It turns out that applying his transformation to our perfect zero-knowledge protocol for Graph Isomorphism yields exactly our interactive proof for Graph Non-Isomorphism.

Recently, it was shown that the error probability allowed in the completeness condition of interactive proofs is not essential for the power of interactive proofs [GMS]. On the other hand, the error probability allowed in the soundness condition is essential to the non-triviality of both interactive proofs and zero-knowledge. Interactive proofs with no error on NO-instances equal $NP$ [GMS], while zero-knowledge proofs with no error on NO-instances equal $RP$ [GO, Ore]. 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 non-isomorphism) cannot be avoided, unless something dramatic happens (e.g. $RP=NP$ or Graph Non-Isomorphism is in $NP$, respectively).

Our result that $NP$ is in zero-knowledge, has been extended to all of IP [BGGHMR]. 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 proofs can have zero-knowledge interactive proofs. The proof extends an earlier proof of ours which applied only to bounded IP.

1.6. Conventions

Let $A$ be a set. Then $Sym(A)$ denotes the set of permutations over $A$. When writing $a \in_R A$, we mean an element chosen at random with uniform probability distribution from the set $A$. When writing $\text{Prob}_{a \in_R A}(P(a))$, we mean the probability that $P(a)$ holds when the probability is taken over all choices of $a \in A$ with equal 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 will consider simple (i.e. no parallel edges) undirected graphs, $G(V,E)$. By $V$ we denote the vertex set, and by $E$ the edge set of the graph $G$. $n$ will 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).

*) Recently, it was also shown that such languages are in $AM(2)$ [AH].
1.7. Organization of the Paper

In Section 2 we present zero-knowledge interactive proofs for graph isomorphism and graph non-isomorphism. These protocols serve as good examples to the notions of interactive proofs and zero-knowledge proofs, and to 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 for graph non-isomorphism.

In Section 3 we show how to use any one-way permutation in order to construct a zero-knowledge interactive proof for any language in NP.
2. PROOFS OF GRAPH ISOMORPHISM AND GRAPH NON-ISOMORPHISM

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

Two graphs $G(V,E)$ and $H(V,F)$ are isomorphic if and only if there exist a permutation $\pi \in \text{Sym}(V)$ such that

$$(u,v) \in E \iff (\pi(u),\pi(v)) \in F.$$  

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 $\pi \in R \text{Sym}(V)$ and letting

$$F = \{(\pi(u),\pi(v)) : (u,v) \in E\}.$$ 

The graph isomorphism problem consists of two graphs as input, and one has to determine whether they are isomorphic. The graph isomorphism problem is trivially in NP, is not known to be in Co-NP, and is believed not to be NP-complete. The fastest algorithm known for the problem runs in time $O\left(\exp\left(O\left(\sqrt{n \log n}\right)\right)\right)$ [BKL].

2.1. An Interactive Proof of Graph Non-Isomorphism

In this subsection we exemplify the notion of an interactive proof system by presenting an interactive proof for graph non-isomorphism. The fact that graph non-isomorphism has interactive proofs is interesting as it is not know to be in NP, and thus has not been know previously to have any efficient proofs. Moreover, the existence of an interactive proof for graph non-isomorphism has interesting complexity theoretic consequences.

In the following protocol the prover needs only to 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 $G_2(V,E_2)$.

1) The verifier chooses at random $n$ integers $\alpha_i \in R \{1,2\}$, $1 \leq i \leq n$. The verifier computes $n$ graphs $H_i(V,F_i)$ such that $H_i$ is a random isomorphic copy of $G_{\alpha_i}$. The verifier sends the $H_i$'s to the prover.

2) The prover answers with a string of $\beta_i$'s (each in $\{1,2\}$), such that $H_i(V,F_i)$ is isomorphic to $G_{\beta_i}(V,E_{\beta_i})$.

3) The verifier tests whether $\alpha_i = \beta_i$, for every $1 \leq i \leq n$. If the condition is violated then the verifier rejects; otherwise he accepts.

Theorem 1: Protocol 1 constitutes a (two-move) interactive proof system for Graph Non-Isomorphism.
Proof: If the graphs $G_1$ and $G_2$ are not isomorphic, and both prover and verifier follow the protocol, then the verifier always accepts. If on the other hand, $G_1$ and $G_2$ are isomorphic then, for each $i$, we have $\alpha_i \neq \beta_i$ with probability at least $1/2$, even if the prover does not follow the protocol. The reason being that in case $G_1$ and $G_2$ are isomorphic,

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

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

The above Theorem has interesting implications on the computational complexity of the graph isomorphism problem. Namely,

Corollary 1: Graph Isomorphism is in $(\text{NP} \cap \text{Co-NP})^A$, for a random oracle $A$. Also, Graph Non-Isomorphism can be recognized by a (non-uniform) family of non-deterministic polynomial-size circuits (i.e. non-uniform NP).

Proof: By the Theorem 1, Graph Non-Isomorphism ($\text{GNI}$) is in $\text{IP}(2)$. Using Goldwasser and Sipser’s transformation of $\text{IP}(k)$ protocols to $\text{AM}(k+2)$ protocols, $\text{GNI} \in \text{AM}(4)$. By Babai’s proof of the finite $\text{AM}(\cdot)$ collapse, $\text{GNI} \in \text{AM}(2) \subseteq \text{NP}^A$ for a random oracle $A$. Finally, it has been pointed out by Mike Sipser that $\text{AM}(2)$ is contained in non-uniform NP.

Corollary 2: If Graph Isomorphism is NP-Complete then the polynomial-time hierarchy collapses to its second level.

Proof: Boppana, Hastad and Zachos showed that if $\text{Co-NP} \subseteq \text{IP}(k)$ (for some fixed $k$) then the entire polynomial-time hierarchy collapses to $\text{AM}(2) \subseteq \Pi_2^P$ [BHZ]. Since Theorem 1 states that graph non-isomorphism is in $\text{IP}(2)$, the Corollary follows.

Corollary 2 may be viewed as providing additional support to 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 for graph isomorphism. The fact that graph isomorphism has efficient proofs is apparent, since it is in NP. However, the fact that graph isomorphism can be proved in zero-knowledge, and in particular without demonstrating the isomorphism is interesting.

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

Protocol 2
common input: Two graphs $G_1(V,E_1)$ and $G_2(V,E_2)$.

Let $\phi$ denote the isomorphism between $G_1$ and $G_2$. The following four steps are executed $n$ times, each time using independent random coin tosses.

P1) The prover generates a graph $H$, a random isomorphic copy of $G_1$. This is done by selecting a permutation $\pi \in \mathcal{R} \text{Sym}(V)$, and computing $H(V,F)$ such that $(\pi(u),\pi(v)) \in F$ iff $(u,v) \in E_1$. The prover sends the graph $H(V,F)$ to the verifier.

V1) The verifier chooses at random $\alpha \in \mathcal{R} \{1,2\}$, and sends $\alpha$ to the prover. (Intuitively, the verifier asks the prover to prove to him that $H$ and $G_\alpha$ are indeed isomorphic.)

P2) If $\alpha=1$ then the prover sends $\pi$ to the verifier, else (i.e. $\alpha \neq 1$) the prover sends $\pi \phi^{-1}$. (Note that the case $\alpha \notin \{1,2\}$ is handled as $\alpha=2$.)

V2) If the permutation received from the prover is not an isomorphism between $G_\alpha$ and $H$ then the verifier stops and rejects; otherwise he continues.

If the verifier has completed $n$ 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 is zero-knowledge since whatever the verifier receives is "useless", as 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 $\alpha$'s depending on the graphs presented to him. In such a case it can not be argued that the verifier only receives random isomorphic copies of the input graph. The issue is fairly involved, as we have to defeat a universal quantifier which 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 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 $G_2$. It follows that when asked (in step P2) to present an isomorphism between $H$ and $G_\alpha$, where $\alpha \in \mathcal{R} \{1,2\}$, the prover fails with probability $\geq 1/2$.

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 to 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 program of a probabilistic polynomial-time machine interacting with the prover, specified by the protocol. We will present a probabilistic 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 \( M_{V'} \) is constructive: given an interactive program \( V' \), we use it in order to construct the machine \( M_{V'} \). Machine \( M_{V'} \) uses \( V' \) as a subroutine while placing strings of its choice on all the tapes of \( V' \). 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 \( V' \). Typically, we will try to guess which isomorphism the machine \( V' \) will ask to check. We will construct the graph \( H \) such that we can answer \( V' \) in case we were lucky. The cases in which we failed will be ignored. It is crucial that from the point of view of \( V' \) the case which leads to our success and the case which leads to our 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 \( M_{V'} \). Let \( Q^* \) be a polynomial that bounds the running time of \( V' \). Clearly, on input \( x=(G_1,G_2) \), the interactive machine \( V' \) reads at most \( Q(1|x|) \) bits from its random tape. On input \( G_1 \) and \( G_2 \), the machine \( M_{V'} \) chooses at random a string \( r \in \{0,1\}^q \), where \( q = Q^*(1|x|) \). This string is fixed as the contents of the random tape of \( V' \), and will be used together with \( x = (G_1,G_2) \) in all calls to \( V' \). Machine \( M_{V'} \) puts \( x \) and \( r \) on its record tape, and proceeds in \( n \) rounds as follows (initially \( i=1 \)).

**Round \( i \):**

S1) \( M_{V'} \) chooses at random \( \beta \in \{1,2\} \) and a permutation \( \pi \in \text{Sym}(V) \). It computes \( H(V,F) \) such that \( (\pi(u),\pi(v)) \in F \) if and only if \( (u,v) \in E_{\beta} \). (Note that \( H \) is an isomorphic copy of \( G_{\beta} \).)

S2) Machine \( M_{V'} \) sets \( \alpha = V'(x,r;H_1,\pi_1,...,H_{i-1},\pi_{i-1},H) \). (\( \alpha \) is the message \( V' \) sends on input \( x \) and \( r \) after receiving messages \( H_1,\pi_1,...,H_{i-1},\pi_{i-1},H \).) Without loss of generality, \( \alpha \in \{1,2\} \) (since otherwise it is treated as \( \alpha = 2 \).) There are two cases.

*Case 1:* \( \alpha = \beta \) (lucky for \( M_{V'} \)). Machine \( M_{V'} \) appends \( (H,\alpha,\pi) \) to its record tape, sets \( H_i \leftarrow H, \alpha_i \leftarrow \alpha, \pi_i \leftarrow \pi \), 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 \( M_{V'} \) outputs its record and halts.

February 23, 1988
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 $M_{V^*}$ does equal 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 $H^{(i)}$ denote $H_1, \pi_1, \ldots, H_{i-1}, \pi_{i-1}, H_i$. Since, for each repetition of the $i$-th round, $\beta \in R \{1,2\}$ ($\beta$ is chosen randomly by $M_{V^*}$ independently of anything else), we get

$$\text{Prob}\left[V^*(x,r;H^{(i)}) = \beta\right] = \frac{1}{2}.$$ 

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 polynomial-time. □

Claim 2.2: The probability distribution $M_{V^*}(G_1G_2)$ is identical to the distribution $<P, V^*>(G_1G_2)$.

Proof: Let $x = (G_1G_2)$. A typical element in the support of $M_{V^*}(x)$ (and $<P, V^*>(x)$) consists of the input $x$, the random tape $r$, and a sequence of $n$ triples $(H_1, \alpha_1, \pi_1), \ldots, (H_n, \alpha_n, \pi_n)$. For $0 \leq i \leq n$, let $\Pi^{(i)}_M$ (resp. $\Pi^{(i)}_{M_{V^*}}$) denote a random variable which results by truncating the random variable $M_{V^*}(x)$ (resp. $<P, V^*>(x)$) after the $i$-th triple. The claim can thus be written as $\Pi^{(i)}_M = \Pi^{(i)}_{M_{V^*}}$, and is proven by induction on $i$.

**Induction Base ($i = 0$):** Clearly, for every $r \in \{0,1\}^9$,

$$\text{Prob}\left[\Pi^{(0)}_M = (x,r)\right] = 2^{-9} = \text{Prob}\left[\Pi^{(0)}_{M_{V^*}} = (x,r)\right].$$

**Induction Step ($i \rightarrow i + 1$):** Let $H^{(i)}$ be an element in the support of $\Pi^{(i)}_M$, and $H_{i+1}, \alpha_{i+1}, \pi_{i+1}$ be a graph, bit and permutation respectively. By induction hypothesis

$$\text{Prob}\left[\Pi^{(i)}_M = H^{(i)}\right] = \text{Prob}\left[\Pi^{(i)}_{M_{V^*}} = H^{(i)}\right]$$

Assuming that $\Pi^{(i)}_M$ (resp. $\Pi^{(i)}_{M_{V^*}}$) equals $H^{(i)}$, we consider the $i+1$st triple in $\Pi^{(i+1)}_M$ (resp. $\Pi^{(i+1)}_{M_{V^*}}$). Such a triple, $(H, \alpha, \pi)$, is determined by $\alpha$ and $\pi$. In fact, the entire triple is determined by $\pi' = \pi \phi^{i-\alpha}$ - the isomorphism between $H$ and $G_1$ ($H$ is clearly determined by $\pi'$ and so is $\alpha = V^*(H^{(i)}H)$). In $\Pi^{(i+1)}_{M_{V^*}}$, we have $\pi' \in R\text{Sym}(V)$, by step (P1). In $\Pi^{(i+1)}_M$, we have $\pi' \leftarrow \pi \phi^{i-\beta}$ where $(\pi, \beta) \in R\text{Sym}(V)\times\{1,2\}$ so that $\beta = f(\pi \phi^{i-\beta})$ with a function $f : \text{Sym}(V) \rightarrow \{1,2\}$. The function $f$ is defined by $f(\pi \phi^{i-\beta}) = V^*(H^{(i)}H)$, where $H$ is the graph resulting by applying the vertex renaming permutation $\pi \phi^{i-\beta}$ to the graph $G_1$ (i.e. applying the vertex renaming permutation $\pi$ to the graph $G_\beta$). It follows that also in $\Pi^{(i+1)}_M$, the permutation $\pi'$ is uniformly distributed over $\text{Sym}(V)$. Thus,

$$\text{Prob}\left[\Pi^{(i+1)}_M = H^{(i)}H_{i+1}\alpha_{i+1}\pi_{i+1} \mid \Pi^{(i)}_M = H^{(i)}\right] = \text{Prob}\left[\Pi^{(i+1)}_{M_{V^*}} = H^{(i)}H_{i+1}\alpha_{i+1}\pi_{i+1} \mid \Pi^{(i)}_{M_{V^*}} = H^{(i)}\right]$$

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

February 23, 1988

Technion - Computer Science Department - Technical Report CS0498 - 1988
Combining the two claims, the Theorem follows.

Remark 10: Serial execution vs. parallel execution: the case where the intuition fails? Although one’s intuition may insist that the above zero-knowledge protocol, remains zero-knowledge even when the $n$ rounds are executed in parallel instead than serially, we do not know how to prove this statement. The reader should note that the argument used in the proof is based on the fact that the verifier’s request for the $i$-th graph only depends on the first $i$ graphs. Thus, changing the $i+1$st graph does not effect 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 $n$ isomorphic copies and only then receives the verifier’s requests. In this case, the verifier’s request for the $i$-th graph may depend on all the other graphs. Trying to satisfy all the requests by choosing the $H_i$’s beforehand, we would have had only a negligible probability of succeeding ($2^{-n}$). Trying 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 $\alpha_i$’s will match these graphs better - may fail too as the $\alpha_i$’s corresponding to the good $H_i$’s may change too. We do not know whether this difficulty is inherent. Our conjecture is that the parallel version is not zero-knowledge.

2.3. Zero-Knowledge Proof of Graph Non-Isomorphism

The interactive proof for graph non-isomorphism presented in section 2.1 is probably not zero-knowledge: a user interacting with the prover may use the prover in order to test to which of the given graphs ($G_1$ and $G_2$) is a third graph $G_3$ isomorphic (for more details see [GO, Ore]). 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.

The resulting (modified) protocol resembles the zero-knowledge proof of Quadratic Non-Residuosity which has appeared before in [GMR]. A unified and generalized presentation appears in [TW]. The presentation of the protocol is further simplified using the "pairing trick" of J. Cohen Benaloh [Bh] (an analouq idea was suggested in [EGL]). For sake of simplicity we replace $G_2$ by $G_0$.

Protocol 3

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

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

VI) The verifier chooses at random $\alpha \in R \{1,0\}$, and a permutation $\pi \in R Sym(V)$. The verifier constructs $H(V,F)$ such that $(\pi(u),\pi(v)) \in F$ iff $(u,v) \in E_\alpha$. (The graph $H$ is a random isomorphic copy of either $G_1$ or $G_0$.) In addition to the graph $H$ the verifier constructs $n^2$ pairs of graphs such that each pair contains one random isomorphic copy of $G_1$ and one random isomorphic copy of $G_0$. That is, for each $1 \leq i \leq n^2$, 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 $j \in \{1,0\}$, $T_{i,j}(V,L_{i,j})$ such that $(\tau_{i,j}(u),\tau_{i,j}(v)) \in L_{i,j}$ iff $(u,v) \in E_{j\gamma_i \mod 2}$. 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 
$$\{(t_i, \tau_{i,1}, \tau_{i,0}) : i \in I\} \text{ and } \{(\alpha + \gamma_i \mod 2, \tau_{i,(\alpha + \gamma_i \mod 2)^{-1}}) : i \in \overline{I}\},$$
where $\overline{I} = \{1, 2, \ldots, n^2\} - I$.

P2) The prover checks if $\tau_{i,j}$ is indeed the isomorphism between $T_{i,j}$ and $G_{(j + \gamma_i \mod 2)^{-1}}$ for every $i \in I$ and $j \in \{1, 0\}$. The prover checks if $\tau_{i,(\alpha + \gamma_i \mod 2)^{-1}}$ is indeed the isomorphism between $T_{i,(\alpha + \gamma_i \mod 2)}$ and $H$, for every $i \in \overline{I}$. If either conditions is violated the prover stops. Otherwise, the prover answers with $\beta \in \{1, 0\}$, such that $H$ is isomorphic to $G_{\beta}$. (If no such $\beta$ 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 $n$ iterations of the above steps then he accepts.

When verifying that the modified protocol still constitutes an interactive proof system for graph non-isomorphism, we have to show that the information revealed by the verifier in step (V2) does not yield information about $\alpha$ (in case the graphs are isomorphic!). When proving that protocol 3 is zero-knowledge, we will use the information revealed in step (V2) in order to help the simulator.

**Theorem 3:** Protocol 3 constitutes a (perfect) zero-knowledge interactive proof system for Graph Non-Isomorphism.

**Proof:** First we prove that Protocol 3 (which is a modification of Protocol 1) is still an interactive proof of graph non-isomorphism. Clearly, the completeness condition still holds (i.e. if the graphs $G_1$ and $G_2$ are not isomorphic, and both prover and verifier follow the protocol, then the verifier always accepts). If on the other hand, $G_1$ and $G_2$ are isomorphic then $\alpha \neq \beta$ with probability at least 1/2, even if the prover does not follow the protocol. The reason being that in case $G_1$ and $G_2$ are isomorphic, the graphs and permutations sent by the verifier in steps (V1) and (V2) are random isomorphic copies of both graphs and yield no information on $\alpha$. Repeating the five steps $n$ times we conclude that the probability that the verifier does not reject two isomorphic graphs is thus at most $2^{-n}$.

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 will present a machine $M_{V^*}$ for every interactive machine $V^*$, so 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. The interested reader is referred to [GMR, TW].)

Following is a detailed description of $M_{V^*}$. 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 $G$. $M_{V^*}$ places $r$ on its record tape and proceeds in $n$ rounds as follows. ($i < -1$)

**Round $i$:**

1. **S1)** $M_{V^*}$ initiates $V^*$ on input $G_1 G_2$ and random tape $r$, and reads the graphs $H$, and $(T_{1,1} T_{1,0}) \ldots (T_{n,1} T_{n,0})$ from the communication tape of $V^*$. (I.e. $H$, $(T_{1,1} T_{1,0}) \ldots (T_{n,1} T_{n,0}) \leftarrow V^*(G_1 G_2 r)$.) $M_{V^*}$ chooses a random subset $I$ and places it on the communication tape of $V^*$. $M_{V^*}$ also appends $I$ to its record tape.

2. **S2)** $M_{V^*}$ reads $\{\langle \gamma_i, \tau_{i,1}, \tau_{i,0} \rangle : i \in I \}$ and $\{\langle (\alpha+\gamma_i \text{ mod } 2, \tau_{i,(\alpha+\gamma_i \text{ mod } 2)^{-1}}) : i \in \overline{I} \}$, from the communication tape of $V^*$. (I.e. $\{\langle \gamma_i, \tau_{i,1}, \tau_{i,0} \rangle : i \in I \}$, $\{\langle (\alpha+\gamma_i \text{ mod } 2, \tau_{i,(\alpha+\gamma_i \text{ mod } 2)^{-1}}) : i \in \overline{I} \} \leftarrow V^*(G_1 G_0 r ; I)$.) $M_{V^*}$ checks if $\tau_{i,j}$ is indeed the isomorphism between $T_{i,j}$ and $G (j+\gamma_i \text{ mod } 2)$, for every $i \in I$ and $j \in \{1,0\}$. It also checks if $\tau_{i,(\alpha+\gamma_i \text{ mod } 2)^{-1}}$ is indeed an isomorphism between $T_{i,(\alpha+\gamma_i \text{ mod } 2)}$ and $H$, for every $i \in \overline{I}$. If either condition is violated $M_{V^*}$ outputs its record tape and stops. Otherwise, $M_{V^*}$ continues to step (S3).

3. **S3)** The purpose of this step is to find an isomorphism between $H$ and $G_\alpha$. 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 $G_1 G_0$, (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 $\{\langle \delta_i, \sigma_{i,1}, \sigma_{i,0} \rangle : i \in K \}$ and $\{\langle (\alpha+\delta_i \text{ mod } 2, \sigma_{i,(\alpha+\delta_i \text{ mod } 2)^{-1}}) : i \in \overline{K} \}$, from the communication tape of $V^*$. (I.e. $\{\langle \delta_i, \sigma_{i,1}, \sigma_{i,0} \rangle : i \in K \}$, $\{\langle (\alpha+\delta_i \text{ mod } 2, \sigma_{i,(\alpha+\delta_i \text{ mod } 2)^{-1}}) : i \in \overline{K} \} \leftarrow V^*(G_1 G_0 r ; K)$.)

   (S3.2) For every $i \in I \cap K$, if $\sigma_{i,(\alpha+\delta_i \text{ mod } 2)^{-1}}$ is indeed the isomorphism between $T_{i,(\alpha+\delta_i \text{ mod } 2)}$ and $H$ then $\tau_{i,(\alpha+\delta_i \text{ mod } 2)^{-1}} \sigma_{i,(\alpha+\delta_i \text{ mod } 2)^{-1}}$ is an isomorphism between $G_\alpha$ and $H$. Similarly, for $i \in \overline{I} \cap K$. If an isomorphism was not found goto step (S3.1).

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

4. **S4)** Once the isomorphism between $H$ and $G_\beta$, $\beta \in \{1,0\}$, is found, machine $M_{V^*}$ appends $\beta$ to its record tape, thus completing round $i$.

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

February 23, 1988
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 $M_{V^*}$ 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 $T_i, T_{i,0}$ and $G_1 G_0$ for $\forall i \in I$ and an isomorphism between one of the $T_i, T_{i,0}$'s and $H$ for $\forall i \in \bar{I}$.) Denote by $g_r$ the number of good subsets. Clearly, $0 \leq g_r \leq 2^{n^2}$. We will compute the expected number of times $V^*$ is invoked in round $i$ 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 another subset $K$ is also good. In case the set chosen in step (S1) is bad, the round is completed immediately. Thus, the expected number of invocations is

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

**Case 2 ($g_r = 1$):** With exponentially small probability (i.e. $2^{-n^3}$) 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 $n! \cdot 2^{-n^3} < 1$ 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 $M_{V^*}(G_1 G_2)$ is identical to the distribution $\langle P, V^* \rangle(G_1 G_2)$.

**Proof:** Both distributions consists of a random $r$, and sequence of elements, each being either $(I, \beta)$ (with good $I$) or a bad $I$, where $I$ is a random subset of $\{1, 2, \ldots, n^2\}$. □

The Theorem follows. ■

**Remark 11:** The validity of Theorem 3 does not rely on the fact that the iterations are executed one after the other. In other words, the five steps protocol in which everything is performed in parallel is also a zero-knowledge interactive proof for Graph Non-Isomorphism. The simulator will also handle all iterations concurrently, first by choosing a subset $I_i$ for each iteration $i$, and next choose subsets $K_i$ for each iteration $i$. The fact that Protocol 3 can be parallelized follows from the fact that its proof technique only needs "something good" (choosing a good $I_i$) to happen once per each iteration (and not necessarily concurrently!). In the proof of Theorem 2 we need "something good" (choosing a lucky $\beta_i$) to happen concurrently in all iterations.
3. ALL LANGUAGES IN NP HAVE ZERO-KNOWLEDGE PROOF SYSTEMS

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

In this section, we assume the existence of secure encryption schemes (in the sense of Goldwasser and Micali [GM] - see definition below). Such schemes exist assuming the existence of one-way permutations (*) [GM, Y1]. A concrete and more efficient implementation, assuming the intractability of factoring, appears in [ACGS].

In this work, 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. For more details consult [GM].

**Definition 6 (Secure Encryption [GM]):** An encryption function is a polynomial-time computable two-argument function $f: \{0,1,2,3\} \times \{0,1\} \rightarrow \{0,1\}$ 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 $r \in \mathcal{R} \{0,1\}^n$.

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 circuits $(C_n)_n$, where $C_n$ is a circuit of size $Q(n)$, every constant $c > 0$ and sufficiently large $n$

$\mid \text{Prob}[C_n(f_n(x)) = 1] - \text{Prob}[C_n(f_n(y)) = 1] \mid < \frac{1}{n^c}$

3.1 A Zero-Knowledge Proof for Graph 3-Colourability

A graph $G(V,E)$ is said to be 3-colourable if there exists a mapping $\phi: V \rightarrow \{1,2,3\}$ (called a proper colouring) such that every two adjacent vertices are assigned different colours (i.e. $\forall (u,v) \in E$ satisfies $\phi(u) \neq \phi(v)$). The language 3-colourability consists of the set of undirected graphs which are 3-colourable. 3-colourability is known to be NP-complete [K].

The common input to the following protocol is a graph $G(V,E)$, which we assume without loss of generality to be simple (i.e. no multiple edges and no self-loops) and connected. In the following protocol, the prover needs only to be a probabilistic polynomial-time machine which gets a proper 3-colouring of $G$ as an input.
auxiliary input. Let us denote this colouring by \( \phi : V \rightarrow \{1,2,3\} \). Let \( n=|V| \), \( m=|E| \) and \( S_3=\text{Sym}(\{1,2,3\}) \). 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 \( V=\{1,2,\ldots,n\} \).

**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 permutation of the 3-colouring, encrypts it, and sends the encryption to the verifier. More specifically, the prover chooses a permutation \( \pi \in \pi \{1,2,3\} \), and random \( n \)-bit \( r_v \)'s (for \( \forall v \in V \), let \( r_v \in \{0,1\}^n \)). 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 \( e \in E \) and sends it to the prover. (Intuitively, the verifier asks to examine the colouring of the endpoints of \( e \in E \).)

**P2)** If \( e=(u,v) \in E \) then the prover reveals the colouring 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 \( E \).

**V2)** The verifier checks the "proof" provided in step (3). 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 3-colourability.

**Proof:** First we show that Protocol 4 constitute an interactive proof for 3-colourability. Clearly, if the graph is 3-colourable and both prover and verifier follow the protocol then the verifier accepts with probability 1. If the graph is not 3-colourable and the verifier follows the protocol then, no matter how the prover plays, at each round the verifier will reject with probability at least \( 1/m \) (use the fact that there is a unique decryption!). 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 \( (1-m^{-1})^{m^2} = \exp(-m) \).

Our demonstration that Protocol 4 is indeed zero-knowledge follows the lines of the proof of Theorem 2. Namely, the construction of \( M_{\phi} \) uses the same underlying ideas used in the proof of Theorem 2. However, additional technical difficulties arise in proving the validity of the construction.

February 23, 1988
As in the proof of Theorem 2, we will present a machine $M_{V^*}$ for every interactive machine $V^*$. This time, $M_{V^*}(x)$ and $<P,V^*>_<(x)$ will not be equal, but instead it will be shown that they are polynomially-indistinguishable. Typically, we will try to guess which edge the machine $V^*$ will ask to check. We will encrypt an illegal colouring 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 which leads to our success and the case which 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. 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 $r \in \{0,1\}^q$ 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 $(u,v) \in E$ and a pair of integers $(a,b) \in \{(i,j): 1 \leq i \neq j \leq 3\}$ at random. $M_{V^*}$ chooses random $r_i$'s ($r_i \in \{0,1\}^*$) and computes $F_i = f(c_i,r_i)$, where $c_i$ is 0 for $i \in V - \{u,v\}$, $c_u = a$ and $c_v = b$.

S2) Machine $M_{V^*}$ sets $e = V^*(G,r;R_1,R_1',...,R_{i-1},R_{i-1}',R_i)$. ($e$ is the message that $V^*$ sends on input $G$ and random tape $r$ after receiving messages $R_1,R_1',...,R_{i-1},R_{i-1}',R_i$. $R_j$ is the $j$th sequence of encrypted colours, and $R'_j$ is the $j$th 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 $R_i \leftarrow (F_1,F_2,...,F_n)$, $e_i \leftarrow e$, $R_i' \leftarrow ((c_u,r_u),(c_v,r_v))$, 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 rounds are completed then $M_{V^*}$ outputs its record and halts.

We now have to prove the validity of the construction. First, we prove that the simulator $M_{V^*}$ indeed 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 which "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 probably 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 which asserts, as a special case, that the "encryption" of the colouring 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.

February 23, 1988
[GM], and its proof is sketched here for the sake of self-containment.

Notations: Let \( a = (a_0, \ldots, a_k) \), where \( a_i \in \{0, 1, 2, 3\} \) for all \( 1 \leq i \leq k \). Then \( \overline{f_n}(a) = f_n(a_1) \cdots f_n(a_k) \).

Lemma 4.0: Let \( P \) be a polynomial, and \( a^{(1)}, a^{(2)}, a^{(3)}, \ldots \) and \( b^{(1)}, b^{(2)}, b^{(3)}, \ldots \) be two infinite sequences such that \( a^{(n)} = (a_0^{(n)}, \ldots, a_k^{(n)}) \) and \( b^{(n)} = (b_0^{(n)}, \ldots, b_k^{(n)}) \), where \( a^{(n)}, b^{(n)} \in \{0, 1, 2, 3\} \) and \( k_n < P(n) \). Then for every polynomial \( Q \), every family of (\( Q \) size) Boolean circuits \( \{C_n : \text{size}(C_n) \leq Q(n)\}_n \), every constant \( c > 0 \) and sufficiently large \( n \)

\[
\left| \Pr[C_n(\overline{f_n}(a^{(n)})) = 1] - \Pr[C_n(\overline{f_n}(b^{(n)})) = 1] \right| < \frac{1}{n^c}
\]

Proof's sketch: The statement is proven considering encryptions of hybrids which consists of a prefix taken from \( a^{(n)} \) and a suffix taken from \( b^{(n)} \). A polynomial-size circuit family that distinguishes the encryptions of the extreme hybrids (i.e. \( a^{(n)} \) and \( b^{(n)} \)) also distinguishes encryptions of two adjacent hybrids. Incorporating the hybrid messages into the circuits one can construct a family of (polynomial-size) circuits which distinguishes the encryptions of \( x \neq y \in \{0, 1, 2, 3\} \). This contradicts our assumption that \( f \) is a nonuniformly secure encryption function.

Corollary 4.0: Let \( P \) be a polynomial, and \( X^{(1)}, X^{(2)}, X^{(3)}, \ldots \) and \( Y^{(1)}, Y^{(2)}, Y^{(3)}, \ldots \) be two infinite sequences such that \( X^{(n)} \) and \( Y^{(n)} \) are independent random variables assuming as values sequences of length \( \leq P(n) \) with elements in \( \{0, 1, 2, 3\} \). Then for every polynomial \( Q \), every family of (\( Q \) size) Boolean circuits \( \{C_n : \text{size}(C_n) \leq Q(n)\}_n \), every constant \( c > 0 \) and sufficiently large \( n \)

\[
\left| \Pr[C_n(\overline{f_n}(X^{(n)})) = 1] - \Pr[C_n(\overline{f_n}(Y^{(n)})) = 1] \right| < \frac{1}{n^c}
\]

Proof: Immediate from Lemma 4.0, by an averaging argument.

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

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 \( i \)-th round, machine \( M_{V^*} \) places the encryption of \( \text{msg}^{(u, v), (a, b)} \) on the communication tape of \( V^* \), where \((u, v) \in R \) and \( a \neq b \in R \{1, 2, 3\} \). Using Corollary 4.0, we will show that the probability that \( V^* \) asks to reveal the colours of \( u \) and \( v \) is essentially equal to the probability \( V^* \) would ask so when interacting with \( P \) (i.e. seeing the encryption of \( \text{msg}^{(\phi, \pi)} \) on its communication tape). It follows that a repetition of round \( i \) is completed successfully with probability \( = 1/m \). A detailed proof follows.

We use \( V^*({\text{enc}}) \) as a shorthand for \( V^*(G, r, R^{(i-1)}, \text{enc}) \). The mapping \( \phi : V \rightarrow \{1, 2, 3\} \) is a fixed 3-colouring of \( G \) used by the prover. First, we bound the probability that round \( i \) is completed successfully. This probability equals the probability that \( \overline{f}(\text{msg}^{(u, v), (a, b)}) = (u, v) \), where \((u, v) \in R \) and
\( a \neq b \in R \{1,2,3\} \).

\[
\Pr_{(u,v) \in E, a \neq b \in R \{1,2,3\}} \left[ V^* \left( f_{\mu}^n (msg_{GM}^{u,v}(a,b)) \right) = (u,v) \right]
\]
\[
= \sum_{(u,v) \in E} \frac{1}{m} \cdot \Pr_{a \neq b \in R \{1,2,3\}} \left[ V^* \left( f_{\mu}^n (msg_{GM}^{u,v}(a,b)) \right) = (u,v) \right]
\]
\[
> \sum_{(u,v) \in E} \frac{1}{m} \left[ \Pr_{\pi \in S_3} \left[ V^* \left( f_{\mu}^n (msg_{GM}^{u,v}(a,b)) \right) = (u,v) \right] - \frac{1}{n^c} \right] 
\]
\[
> \frac{1}{m} \cdot \sum_{(u,v) \in E} \left[ \Pr_{\pi \in S_3} \left[ V^* \left( f_{\mu}^n (msg_{GM}^{u,v}(a,b)) \right) = (u,v) \right] - \frac{1}{2m} \right]
\]
\[
= \frac{1}{m} \left[ \Pr_{\pi \in S_3} \left[ V^* \left( f_{\mu}^n (msg_{GM}^{u,v}(a,b)) \right) \in E \right] - \frac{1}{2} \right] 
\]
\[
= \frac{1}{2m}
\]

In passing from the second line to the third one, the inequality holds since otherwise \( V^* \) and \( R^{(1)} \) can be incorporated into a circuit that distinguishes the encryptions of \( msg_{GM}^{u,v}(a,b) \) and \( msg_{GM}^{u,v}(\pi) \) (for some \( u,v,a,b \) and \( \pi \)), in contradiction to Lemma 4.0. (In passing from the third line to the fourth one, we use \( m < n^2/2 \).)

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. \( \square \)

**Claim 4.2:** Let \( 3C \) denote the set of 3-colourable graphs. Then the probability ensembles \( \{M_{V^*}(G)\}_{G \in 3C} \) and \( \{(P,V^*)(G)\}_{G \in 3C} \) are polynomially-indistinguishable.

**Proof:** The proof is by contradiction. We assume that the two probability ensembles can be told apart by polynomial-size circuit and derive contradiction to Lemma 4.0 (or Corollary 4.0) in one of essentially two ways. First, we restrict our attention to a single round of the \( (P,V^*) \) conversation and its simulation. Next, we consider two cases. In case the distribution of the "verifier's query" is substantially different in the conversation and the simulation the program \( V^* \) can be used to contradict Corollary 4.0. In case 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 receives as input a text which is not of the form \( f_{\mu}^n (\alpha) \), for a sequence \( \alpha \), but rather 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 some difficulties which need to be resolved with some care.

Let \( \{C_n\}_n \) be a circuit family of polynomial size which distinguishes \( \{M_{V^*}(G)\}_{G \in 3C} \) from \( \{(P,V^*)(G)\}_{G \in 3C} \). 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 \( i \)
Let us consider this circuit family. For sake of simplicity, we carry our argument while referring to an arbitrary graph, denoted \( G(V,E) \), in the above sequence of graphs and to the corresponding circuit \( C_{1V_1} \). 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 \( \phi \) denote the fixed colouring of \( G(V,E) \) used by the prover, \( n = |V| \) and \( m = |E| \).

A typical element in the support of \( M_{V'}(G) \) (resp. in the support of \( \phi.V'(G) \)) consists of the input graph \( G \), the random tape \( r \in \{0,1\}^q \), and a sequence of \( m^2 \) triples \( (R_1,e_1,R'_{i}) \ldots (R_{m^2},e_{m^2},R'_{m^2}) \). For \( 0 \leq i \leq m^2 \), let \( \Pi(i) \) denote the \( i \)-th hybrid distribution defined by taking \( G, r \in \{0,1\}^q \) and the first \( i \) triples \( (R_1,e_1,R'_{i}) \ldots (R_i,e_i,R'_{i}) \) from the distribution \( \langle P,V' > (G) \), 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 \( R_1,R'_{i} \ldots R_i,R'_{i} \). Note that \( \Pi(0) \) equals \( M_{V'}(G) \), while \( \Pi(m^2) \) equals \( \langle P,V' > (G) \).

Clearly, the circuit \( C_n \) which distinguishes the extreme hybrids (i.e. \( \Pi(0) \) and \( \Pi(m^2) \)) also distinguishes some two neighbouring 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

\[
\Pr\left[ C_n(\Pi(i)) = 1 \right] - \Pr\left[ C_n(\Pi(i+1)) = 1 \right] > \frac{1}{m^2.n^\epsilon}
\]

Let \( \epsilon = \frac{1}{m^2.n^\epsilon} \). Let \( \Pi(i,j) \) be the distribution resulting 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. \( \epsilon \)) and the polynorniality of the size of its members, so that it distinguishes \( \Pi(i,j) \) and \( \Pi(i+1,j+1) \). (This is done by incorporating \( M_{V'} \) into the circuits and applying it to construct the last \( m^2-(i+1) \) pairs.)

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

\[
\Pr\left[ C_n(\Pi_M(H_i)) = 1 \right] - \Pr\left[ C_n(\Pi_{PV}(H_i)) = 1 \right] > \epsilon \left[ = \frac{1}{m^2.n^\epsilon} \right]
\]
Recall that \( \phi : \{1, 2, \ldots, n\} \rightarrow \{1, 2, 3\} \) is a fixed colouring of \( G \). Note that \( \Pi_{PV}(H_i) \) equals \( (f_n(msg_{IP})(u, v), ((\pi(\phi(u)), r_u), (\pi(\phi(v)), r_v))) \), where \( \pi \in R S_3 \), \( (u, v) = V^*(G, r; H_i, f_n(msg_{IP})) \) and \( r_u \) (resp. \( r_v \)) is the randomization used to encrypt the \( u \)-th (resp. \( v \)-th) element in \( msg_{IP} \). On the other hand \( \Pi_M(H_i) \) equals \( (f_n(msg_{SM}(u, v)), (a, b), (a, r_u), (b, r_v)) \), where \( u, v, a, b \) are chosen by some distribution, \( (u, v) = V^*(G, r; H_i, f_n(msg_{SM}(u, v))) \) and \( r_u \) (resp. \( r_v \)) is the randomization used to encrypt the \( u \)-th (resp. \( v \)-th) element in \( msg_{SM}(u, v), (a, b) \). A typical element in both distributions is an encryption of a \( n \)-element vector and two pairs corresponding to the decryption of two of its entries (the \( u \)-th and \( v \)-th entry).

We derive a contradiction (to Lemma 4.0) by using the circuit \( C_n \) to construct a circuit \( C_n' \), of polynomial size, which 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. \( msg_1 = 0^n 3^n \) and \( msg_2 = 1^n 2^n 3^n \)).

**The Construction of \( C_n' \):**

The circuit \( C_n' \), incorporating the circuit \( C_n \), the colouring \( \phi \), the (verifier’s) program \( V^* \) and the history \( H_j \), operates as follows. On input a text \( t_1t_2 \ldots t_{3n} \), where \( t_i \in \{0, 1\} \), the circuit \( C_n' \) selects at random \( (u, v) \in R E \), \( \pi \in R S_3 \) and \( r_u, r_v \in R \{0, 1\}^n \). The circuit \( C_n' \) sets \( c_j \leftarrow (\pi(\phi(u))) \) and \( i_j \leftarrow (c_j - 1) \cdot n + j \) and constructs

\[
\text{text} \leftarrow t_{i_1}t_{i_2} \ldots t_{i_n}, f(c_u, r_u)_{i_n}, \ldots, f(c_v, r_v)_{i_n}, \ldots, t_{i_n}
\]

The circuit \( C_n' \) then runs \( V^* \) on \( H_i \) and \( \text{text} \). If \( V^*(H_i, \text{text}) = (u, v) \) then \( C_n' \) stops outputting \( 0 \). If \( V^*(H_i, \text{text}) = (u, v) \) then \( C_n' \) applies \( C_n \) and stops outputting \( C_n(\text{text}, (u, v), (\pi(\phi(u)), r_u), (\pi(\phi(v)), r_v)) \).

We will be interested in the behaviour of the circuit \( C_n' \) on the input distributions \( \tilde{f}_n(msg_1) \) and \( \tilde{f}_n(msg_2) \). Let \( D_1 \) (resp. \( D_2 \)) denote the probability distribution of \( \text{text} \), constructed by \( C_n' \), on input distribution \( \tilde{f}_n(msg_1) \) (resp. \( \tilde{f}_n(msg_2) \)). The distribution \( D_1 \) (resp. \( D_2 \)) depends on \( (u, v), \pi, r_u, r_v \) 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 \( \tilde{f}_n(msg_1) \) (resp. \( \tilde{f}_n(msg_2) \)). 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 \( 0 \), \( f(\pi(\phi(v)), r_v) \) 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 \( f(\pi(\phi(u)), \ldots, f(\pi(\phi(u))), \ldots, f(\pi(\phi(u))), \ldots, f(\pi(\phi(u))] \), the element \( f(\pi(\phi(u)), r_u) \), the distribution \( f(\pi(\phi(u))), \ldots, f(\pi(\phi(v))), \ldots, f(\pi(\phi(v))] \), the element \( f(\pi(\phi(v)), r_v) \), and the distribution \( f(\pi(\phi(v))), \ldots, f(\pi(\phi(v))] \). 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:**

\[ \left| \text{Prob} \left[ V^*(H_i, D_1^{(u,v), \pi, r_u, r_v}) = (u, v) \right] - \text{Prob} \left[ V^*(H_i, D_2^{(u,v), \pi, r_u, r_v}) = (u, v) \right] \right| < \frac{e}{2m} \]

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.

February 23, 1988
Proof: Assuming on the contrary that Claim 4.2.1 does not hold, we derive contradiction to Corollary 4.0 by constructing a circuit (incorporating $V^*$ and $H_i$) that distinguishes $D_1$ from $D_2$ (which are probabilistic encryptions of $n$-element sequences).

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

Claim 4.2.2:

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

Proof: Let $\xi$ be a random variable assuming values in $E \times S_3 \times \{0,1\}^n \times \{0,1\}^n$. By $D_1^\xi$ (resp. $D_2^\xi$) we shorthand $D_1^{(u,v),\pi_{x,y}}$ (resp. $D_2^{(u,v),\pi_{x,y}}$), where the probability is taken over all choices of $(u,v) \in E$, $\pi \in S_3$, and $r_{u,v} \in \{0,1\}^n$, with uniform probability distribution.

By the construction of $C'_n$ we have

$$\Pr \left[ C'_n (\tilde{f} (msg_1)) = 1 \right] = \Pr \left[ V^* (H_i, D_1^\xi) = (u,v) \right] \cdot \Pr \left[ C_n (D_1^\xi, \xi') = 1 \mid V^* (H_i, D_1^\xi) = (u,v) \right]$$

(1)

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

Similarly,

$$\Pr \left[ C'_n (\tilde{f} (msg_2)) = 1 \right] = \Pr \left[ V^* (H_i, D_2^\xi) = (u,v) \right] \cdot \Pr \left[ C_n (D_2^\xi, \xi') = 1 \mid V^* (H_i, D_2^\xi) = (u,v) \right]$$

(2)

By Claim 4.2.1, we have

$$\left| \Pr \left[ V^* (H_i, D_1^\xi) = (u,v) \right] - \Pr \left[ V^* (H_i, D_2^\xi) = (u,v) \right] \right| < \frac{\epsilon}{2m}$$

(3)

By observing that $D_2^{(u,v),\pi_{x,y}}$ is independent of the choice of $(u,v) \in E$ as long as $\pi \in R S_3$ and $r_{u,v} \in R \{0,1\}^n$, we get

$$\Pr \left[ V^* (H_i, D_2^\xi) = (u,v) \right] = \frac{1}{m}$$

(4)

and

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

(5)

(using also the structure of $<P, V^*>$.) By the construction of the simulator, $M_{\nu^*}$, we have

$$\Pr \left[ C_n (D_2^\xi, \xi') = 1 \mid V^* (H_i, D_2^\xi) = (u,v) \right] = \Pr \left[ C_n (\Pi_{\nu} (H_i)) = 1 \right]$$

(6)

Combining (1-6) we get

$$\left| \Pr \left[ C'_n (\tilde{f} (msg_1)) = 1 \right] - \Pr \left[ C'_n (\tilde{f} (msg_2)) = 1 \right] \right| > \frac{1}{m} \cdot \Delta - \frac{\epsilon}{2m}$$

(7)

where $\Delta$ is the difference between $\Pr \left[ C_n (\Pi_{\nu} (H_i)) = 1 \right]$ and $\Pr \left[ C_n (\Pi_{PV} (H_i)) = 1 \right]$. Recalling that
\( \Delta > \varepsilon \) and using (7), Claim 4.2.2 follows. 

Once proved, Claim 4.2.2 contradicts Lemma 4.0, and thus our hypothesis that \((M_{V'}(G))_{G \in 3C}\) and \(\langle P, V' \rangle(G)_{G \in 3C}\) are polynomially distinguishable is contradicted. Claim 4.2 follows. 

Combining Claims 4.1 and 4.2, the Proposition follows. 

Remark 12: The above protocol needs \(m^2\) rounds. There are two alternative ways of modifying the above protocol so to get a four-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 coloured graph) before the prover sends to the verifier the corresponding coloured 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 [GMRiv] 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 a pseudoproof (see section 1.4), is natural in the cryptographic applications. The details are currently being worked out by Oded Goldreich and Ariel Kahn.

Remark 13: Protocol 4 is zero-knowledge assuming that the encryption function used is non-uniformly secure. We do not know whether Protocol 4 remains zero-knowledge even 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 input (graph) on which the simulator fails. The details are beyond the scope of this paper.

3.2 Zero-Knowledge Proofs for all NP

The following theorem asserts that every language \(L \in NP\) 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-colourability (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-colourable. 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 subroutine simulation (and thus is "auxiliary input zero-knowledge" [GO,
The third alternative (due to M. Blum) of proving the theorem consists of presenting a zero-knowledge proof for \( L \) directly, i.e. using the non-deterministic circuit recognizing \( L \). Here we present a proof following 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 [GJ].) Let \( L \in NP \), and \( t \) be the polynomial-time reduction of \( L \) to 3-Colourability (3C). 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 3C (presented in [GJ]). Recall that \( x \in L \) iff \( t(x) \) is 3-colourable. A zero-knowledge interactive proof 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 to prove that \( G \) is 3-colourable. The verifier acts according to the result of this subprotocol.

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

Let \( V^* \) be an ITM which interacts with \( P \), the prover specified for \( L \). Note that \( P \) applies algorithm \( t \) to the input \( x \) and initiates \( P_{3C} \), the prover specified for 3C, on \( t(x) \). We consider the ITM \( V^{**} \) which interacts with \( P_{3C} \) on input \( t(x) \) and auxiliary input \( x \) (i.e. \( <P_{3C},V^{**}(x)>\langle t(x) \rangle = <P,V^*(x)>\langle x \rangle \)). Note that \( V^{**} \), which has an "auxiliary input" is not of the form allowing the application of the fact that \( P_{3C} \) 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 invertable.) Now, by the fact that \( P_{3C} \) is a zero-knowledge prover it follows that there exists a \( M_{V^{***}} \) such that \( \{<P_{3C},V^{***}>\langle t(x) \rangle \}_{x \in L} \) and \( \{M_{V^{***}}(t(x)) \}_{x \in L} \) are polynomial-indistinguishable. Let \( M_{V^*}(x) = M_{V^{***}}(t(x)) \). Since \( <P,V^*>(x) \) equal \( <P_{3C},V^{***}>(t(x)) \), the theorem follows. \( \blacksquare \)

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 transform efficiently also the solution 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 non-deterministic choices leading a fixed NP machine to accept \( x \). Formally, let \( P_L : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\} \) be a polynomial-time computable predicate such that \( x \in L \) iff \( \exists y \) s.t. \( P_L(x,y) = 1 \). Then \( y \) is called an NP proof of "\( x \in L \)."

February 23, 1988

Technion - Computer Science Department - Technical Report CS0498 - 1988
Proof: Let \( t \) be as in the proof of theorem 5. Let \( t' \) be a polynomial-time, computable function transforming the sequence of non-deterministic choices leading to acceptance of \( x \) into a proper 3-colouring of the graph \( t(x) \). (i.e. if \( P_L(x,y) = 1 \) then \( t'(y) \) is a 3-colouring 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\(^*\). Nevertheless, the standard reduction of \( L \) to \( 3C \), has this property. Recall that the prover in Protocol 4 may be a probabilistic polynomial-time machine that gets a 3-colouring 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-colouring \( \phi \) of \( G \).) The Theorem follows.$$

Remark 14: 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 \( y \) having additional information \( z \), can be efficiently computable from \( x \) and \( z \) (without interacting with anybody). Here the auxiliary input is to the verifier which even with it in his possession can not 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 [GO, Ore]. The reason that this extension is important to cryptographic applications is that in a typical application zero-knowledge proofs are used as subprotocols inside a larger 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 15: Another issue which is of importance for practical applications is the efficiency of a zero-knowledge proof, and in particular the efficiency of a zero-knowledge proof for a language \( L \) as function of the non-deterministic Turing machine complexity of \( L \). There are two standard efficiency measures which may be considered:

1) The computational complexity of the proof (i.e. number of steps taken by either or both parties).

2) The communication complexity of the proof. Here one may consider the number of interactions, and/or the total number of bits exchanged. (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 non-standard measure for the efficiency of a zero-knowledge proof is its tightness. Intuitively, tightness is the ratio between the time it takes the simulator to simulate an interaction with the prover over the time the interaction takes the verifier. Formally, the knowledge-tightness (tightness) of a zero-knowledge proof is its tightness

\(^*\) Interestingly, polynomial-time computation of the witness transformation is guaranteed by Levin’s definition of NP-completeness [L].
function \( f: N \rightarrow \mathbb{Z} \) (from integers to integers) satisfying the following: for all polynomial-time ITMs \( V^* \) there exists a machine \( M_{V^*} \) such that \( \{ M_{V^*}(x) \}_{x \in L} \) is polynomially indistinguishable from \( \{ <P,V^*>(x) \}_{x \in L} \), and

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

where \( \text{time}(A(x)) \) is the 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 (2 and 3, respectively) 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 refine notions of zero-knowledge and in particular constant-tightness zero-knowledge. Such refine notions may be applied in a non-trivial manner also to languages in \( \text{P} \).

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 polylogarithmic exist for all languages in \( \text{NP} \) (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 which uses the circuit value problem. For more details see [G2].

### 3.3. Applications - an Example

Due 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 which until recently was considered very complex: Verifiable Secret Sharing. The more general implications of Theorem 6, are investigated in [GMW1, GMW2].

The notion of a verifiable secret sharing was presented by Chor, Goldwasser, Micali and Awerbuch [CGMA], and constitutes a powerful tool for multi-party protocol design. Loosely speaking, a verifiable secret sharing is a \( n+1 \)-party protocol through which a sender (S) can distribute to the receivers (\( R_i \)'s) pieces of a secret \( s \) recognizable through an a-priori known ‘“encryption’’ \( g(s) \). 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 [Sha], 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 [CGMA], improvements in efficiency and "tolerance" appeared in [FM, AGY]. 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 (non-verifiable) secret sharing [Sha]. To share a secret \( s \in GF(p) \) recognizable through \( r = g(s) \), the sender proceeds as follows: First, the sender chooses at random a \( l \)-degree polynomial over \( GF(p) \) and evaluates it in \( n \) fixed points (these are the pieces in Shamir's scheme). Next, the sender encrypts the \( i \)th piece using the Public encryption algorithm of the \( i \)th 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 \( g \) to the free term of this polynomial yields \( r \) (note that this is a NP statement).

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

Due to its generality, Theorem 6 has a dramatic effect on the design of cryptographic protocols. It is the key tool in a general methodology for designing secure cryptographic protocols [GMW2]. This methodology consists of efficient "correctness and privacy preserving" transformations of protocols from a weak adversary model to the most adversarial model. In the weak adversary model, the messages sent are always according to the protocol and the only freedom of the adversary is in storing all intermediate steps of the computation. In the strong adversary model, a party can deviate from the protocol by conducting arbitrary polynomial-time computations.

So far NP-completeness 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 proofs of every NP statement) from Proposition 4 (i.e. a zero-knowledge proof of a particular NP-Complete problem).

ACKNOWLEDGEMENTS

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 Ariel Kahn, Hugo Krawczyk, Eyal Kushilevitz, and Yair Oren for remarks concerning earlier versions of this manuscripts.
REFERENCES


February 23, 1988


February 23, 1988