

The NP-Completeness Column: An Ongoing Guide

DAVID S. JOHNSON

AT&T Bell Laboratories, Murray Hill, New Jersey 07974

This is the 21st edition of a (supposedly) quarterly column that covers new developments in the theory of NP-completeness. The presentation is modeled on that used by M. R. Garey and myself in our book "Computers and Intractability: A Guide to the Theory of NP-Completeness," W. H. Freeman & Co., New York, 1979 (hereinafter referred to as "[G&J]"; previous columns will be referred to by their dates). A background equivalent to that provided by [G&J] is assumed, and, when appropriate, cross-references will be given to that book and the list of problems (NP-complete and harder) presented there. Readers who have results they would like mentioned (NP-hardness, PSPACE-hardness, polynomial-time-solvability, etc.) or open problems they would like publicized, should send them to David S. Johnson, Room 2D-150, AT&T Bell Laboratories, Murray Hill, NJ 07974 (or to dsj@research.att.com). Please include details, or at least sketches, of any new proofs; full papers are preferred. If the results are unpublished, please state explicitly that you are willing for them to be mentioned. Comments and corrections are also welcome. For more details, see the December 1981 issue of this Journal.

INTERACTIVE PROOF SYSTEMS FOR FUN AND PROFIT

This column will be devoted to the concept of the "interactive proof system," together with its popular spin-off, the "zero-knowledge" proof. Recent STOC and FOCS conferences have seen an outpouring of papers on these topics, and it will be impossible for me to cover them all. I shall, however, try to sketch what appear today to be the major issues and results, as usual placing special emphasis on their relation to the theory of NP-completeness.

"Interactive proof systems" augment our standard notion of "proof" in two inter-dependent directions. First, there is interaction. In a standard proof, the proof is given and need only be verified. In an interactive proof, the proof arises out of conversation between a "prover" and a "verifier." This interaction is made useful by the second augmentation: randomization. Both prover and verifier are allowed to flip unbiased coins, and base their questions and answers on the results of the flips (as well as the results of their previous flips and the transcript of the conversation so far). At the end, the verifier must be convinced, with "high probability," that the claimed theorem is true. This is somewhat

less certainty than we have come to expect from a “proof,” but not necessarily much less, in practice. Section 1 covers the two main formalizations of the notion of interactive proof (the original “interactive proof system” of Goldwasser, Micali, and Rackoff [23,24] and the “Arthur-Merlin game” of Babai [4]), illustrating them and discussing the complexity classes they determine.

We then turn to the concept of a “zero-knowledge” proof. This is an interactive proof that convinces the verifier of the statement to be proved, but provides no additional information. For instance, with a zero-knowledge proof that a graph is Hamiltonian, you could convince a colleague that the graph contains a Hamiltonian circuit, while providing no help toward finding such a circuit. Sections 3 and 4 cover two main variations on this idea, the “perfect zero-knowledge proof” (Section 3) and the “computational zero-knowledge proof” (Section 4). The idea of zero-knowledge may have far more important applications than the obvious one of tantalizing and frustrating your colleagues, and I shall indicate several in the latter section.

We conclude with an unrelated news flash: Yet another of the open problems from the list of twelve given in [G&J] has been resolved. (There are now only four left.) For the identity of this problem, along with an almost zero-knowledge hint at the proof that resolved it, see Section 4. (Strong evidence as to the status of one of the remaining four, GRAPH ISOMORPHISM, has been found as a surprising corollary to the work on interactive proof systems, and is discussed at the end of Section 2.)

1. MERLIN, ARTHUR, BOB, AND ALICE

The idea of an interactive proof system was originally proposed by Goldwasser, Micali, and Rackoff in [23,24]. In their model, an interactive proof system consists of two communicating Turing machines, each with a private source of random bits (for instance, a semi-infinite input tape filled with random bits that can be read from left to right) and a common input (the statement to be proved). As has become the custom when dealing with two-party protocols, let us call these machines “Alice” and “Bob” (the user-friendly version of “A” and “B”). Alice is the all-powerful “prover.” The only resource bound she must obey is a polynomial bound on the *length* of the messages she sends; she may use as much time and space as she desires in computing the message. Bob, the “verifier,” is more limited: his cumulative computation time must obey a fixed polynomial bound. Alice and Bob alternate sending each other messages (by writing on a special “interaction tape”). Each party is inactive while the other party is reading his or her message and constructing a response. Either Alice or Bob may on their turn announce that the conversation is over, at which point Bob, perhaps after some additional computation, must decide whether or not to accept the statement.

In typical complexity-theoretic fashion, we may view an interactive proof

system as simply a method for recognizing a language. We say that an interactive proof system (Alice,Bob) recognizes a language L if the following two conditions are satisfied. The first is *completeness*: for any string $x \in L$, the probability that Bob halts and accepts x must be at least $2/3$. (A technical note: this probability is with respect to Bob's random bits only. Thus the $2/3$ lower bound must hold for any given string of bits that Alice might receive from *her* random source.) The second condition is *soundness*: for any string $x \notin L$, the probability that Bob accepts must be $1/3$ or less, *even if* Alice is replaced by an imposter whose express goal is to mislead him. (The choice of $2/3$ and $1/3$ is for simplicity; any values bounded away from $1/2$ will do, and by performing repeated executions of the proof system, one can even meet such stringent requirements as $1 - (1/2)^{|x|}$ and $1/2^{|x|}$ respectively.)

Note that, also in typical complexity-theoretic fashion, this definition of language recognition is not symmetric with respect to $x \in L$ and $x \notin L$. Thus, if we let "IP" represent the set of all languages recognized by interactive proof systems, the fact that $L \in \text{IP}$ does not necessarily imply that $\text{co-}L \in \text{IP}$, where $\text{co-}L = \Sigma^* - L$.

Let us now illustrate the concept of an interactive proof system, so that we can get some idea of what kind of languages *are* in IP. As a first (trivial) illustration, note that any language in NP has an interactive proof system: if $x \in L$, Alice simply sends Bob a list of the choices made in an accepting computation of an NDTM that recognizes L , and Bob verifies that an accepting computation does indeed result. Here no random bits are required, and Bob accepts with probability 1 if $x \in L$ and rejects with probability 1 otherwise. We thus have $\text{NP} \subseteq \text{IP}$.

As a second source of trivial examples, consider BPP, i.e., the class of all languages recognizable by random Turing machines whose probability of error is always $1/3$ or less, no matter whether $x \in L$ or $x \notin L$. BPP was introduced by Gill in [17] and was discussed in the [September, 1984] column. It contains P, in a sense extending P to the broadest set of problems that can reasonably be considered "tractable" when randomization is allowed. It appears neither to contain nor to be contained in NP, although it may well be contained in $\text{NP} \cup \text{co-NP}$. Note that any language in BPP is recognizable by an interactive proof system in which Alice is vacuous and Bob reaches his decision without any outside consultation. Thus BPP is also contained in IP.

We can now see why both randomization and interaction are necessary for the concept of interactive proof to yield something new. If we did not allow for the possibility of random bits in interactive proofs, we would simply have $\text{IP} = \text{NP}$. (In this case a proof system for a language L could be replaced by an NDTM that recognizes L by simulating Bob while "guessing" Alice's responses.) If we did not allow for the possibility of interaction, we would simply have $\text{IP} = \text{BPP}$. Given that we do allow both possibilities, we know that, at the very least, IP contains $\text{NP} \cup \text{BPP}$. The next example suggests that the containment is proper.

Consider the language “NON-ISOMORPHIC GRAPHS” corresponding (under some fixed encoding) to the problem of deciding whether two given graphs G_1 and G_2 are non-isomorphic. It is currently open as to whether NON-ISOMORPHIC GRAPHS is in NP or BPP, although it is clearly in co-NP. Here is an interactive proof system for recognizing it, due to Goldreich, Micali, and Wigderson [20,21]: Bob begins by randomly choosing an integer $i \in \{1,2\}$ and sending Alice an isomorph H of the graph G_i , obtained by randomly permuting G_i 's vertex names. Alice then checks to see whether there is a $j \in \{1,2\}$ such that H is isomorphic to G_j , and if so, sends such a j to Bob. (If not, she halts; there is no need to proceed if Bob is cheating.) This procedure is repeated twice, and Bob accepts if $j = i$ on both rounds. Note that if G_1 and G_2 are not-isomorphic, Alice will choose $j = i$ both times, and so Bob accepts with probability 1. If the two graphs are isomorphic, however, there is no way for Alice (or any imposter) to guess the value of i chosen by Bob with probability greater than $1/2$ on either round, so the probability that Bob accepts is at most $1/4$. Thus this proof system is both complete and sound, and can be said to recognize NON-ISOMORPHIC GRAPHS.

Note that in the above proof system, the probability that Bob accepts when $x \in L$ is 1, not simply the $2/3$ or greater that is all that is required by the definitions. When a proof system obeys this stronger bound, we say that it has *perfect completeness*. This would seem to be a substantial restriction, but, surprisingly, it turns out that all languages in IP are recognizable by perfectly complete interactive proof systems [19]. This, however, is more than can be said for the analogous concept of “perfect soundness.” (An interactive proof system has *perfect soundness* if, for all $x \notin L$, the probability that x is accepted is 0.) The generic systems mentioned above for problems in NP have perfect soundness, but these are essentially the only ones that do: Any language that is recognized by a perfectly sound interactive proof system must be in NP [19].

Having defined IP, Goldwasser, Micali, and Rackoff [23] attempted to get a handle on its structure by defining a hierarchy of subclasses within it, $IP[k]$, $k \geq 0$. These classes were based on restrictions on the number of “rounds” required by the interactive proof system. One exchange of messages (Alice followed by Bob or vice versa) is called a *round*. Note that Bob's overall time bound implies that the total number of rounds in an interactive proof system must be polynomially bounded. What if we impose a more stringent requirement? Note, for instance, that the above protocol for GRAPH NON-ISOMORPHISM requires only two rounds, irrespective of the graphs involved, and these two rounds could be reduced to one by running them in parallel. (Bob chooses *two* indices i_1 and i_2 , and sends Alice random permutations of *both* G_{i_1} and G_{i_2} .) Let us define $IP[k]$ to be the set of languages recognizable by interactive proof systems with k or fewer rounds. One might reasonably conjecture that $IP[k] \neq IP[k+1]$ for all $k \geq 0$ (Goldwasser, Micali, and Rackoff did so in [23]), and that the hierarchy $\{IP[k]; k \geq 0\}$ is strictly contained in IP (just as one

conjectures that the polynomial hierarchy is non-collapsing and strictly contained in PSPACE).

Before we address the evidence on these conjectures, however, it is time to consider the alternate, seemingly more restricted version of interactive proof proposed by Babai [4] at the same conference where [23] appeared: the “Arthur-Merlin game.” An Arthur-Merlin game is an interactive proof system (with Alice and Bob going under the pseudonyms of Merlin and Arthur respectively). In it, Merlin and Arthur (Alice and Bob) retain their respective computational powers, but Arthur’s (Bob’s) communication capabilities are severely limited. Now his messages can consist only of sequences of new random bits from his private source. The lengths of these sequences are determined by the input, and do not depend on their content or on the transcript of the conversation so far. Thus Arthur has essentially nothing to do until Merlin declares the conversation to be over, at which point Arthur has polynomial time to reach his decision, based on the original input and the transcript of the conversation (but without the use of any additional random bits). For an Arthur-Merlin game to recognize a language L , precisely the same soundness and completeness criteria must be met as for an interactive proof system.

Babai introduced Arthur-Merlin games because he wished to classify more accurately the complexity of certain problems about matrix groups, in particular, the MATRIX GROUP EXACT ORDER problem. In this problem, one is given a prime power $q = p^n$, integers k and m , and a collection C of $k \times k$ matrices over $GF[q]$, and asked whether the matrix group generated by C has order equal to m , i.e., whether there are precisely m distinct matrices that can be obtained by multiplying together sequences of members of C (where repetitions are allowed and the product of the empty sequence is taken to be the identity matrix). Although telling whether the order is divisible by a given integer is in NP [5], the status of MATRIX GROUP EXACT ORDER remains open.

In order to classify this problem more accurately, Babai defined various complexity classes based on Arthur-Merlin games, just as IP and the IP[k] were defined for interactive proof systems. The analog of IP is “AM[poly],” the set of all languages recognizable by Arthur-Merlin games with a polynomial number of messages. For analogues of the IP[k], Babai introduced a more refined set of classes, depending on which speaker goes first in the conversation and precisely how many messages were sent. Each class is named by an alternating string of A’s and M’s, which identifies the set of messages and the order in which they sent, each message being identified by the initial of the sender, e.g., “MAM” or “AMAMAM.” As a shorthand, Babai let “AM[k]” represent an alternating string of length k that started with an A, and “MA[k]” represent an alternating string of length k that started with an M. Note that MA[1] thus represents M, which is simply another name for NP. Similarly, AM[1] represents A, which is another name for BPP. (In this case, Arthur’s single message is not intended for Merlin, who after all will not have the opportunity to respond. It is instead

intended for Arthur himself: by sending it he puts random bits into the transcript, and these will now be available for use in his own final computation, which would otherwise have to be strictly deterministic.) To complete the low-end picture, $AM[0] = MA[0] = P$.

We begin to get seemingly new classes when we turn to $MA[2] = MA$ and $AM[2] = AM$. Both classes can be viewed as generalizations of NP. The first, MA, might be called NBPP (nondeterministic BPP), since it consists of languages recognizable by “guess and check” algorithms in which the “check” is done by a probabilistic polynomial-time algorithm, rather than simply a deterministic one. The second, AM, generalizes NP in that it contains NP^B for almost all oracles B . (It might even be the case that $AM = NP^B$ for almost all oracles B , although the supposed proof of this claim in [4] is erroneous. The analogous result that $BPP = P^B$ for almost all oracles B is known to be true [6].) Babai showed in [4] that $NP \subseteq MA \subseteq AM$, and that MATRIX GROUP EXACT ORDER was in AM (indeed, in $AM \cap \text{co-AM}$). Thus it would appear that AM might well be a proper superset of NP.

This might lead us to suspect that the entire $AM[k]$ and $MA[k]$ hierarchies might be non-trivial. Surprisingly, this is not the case. In [4], Babai shows that for all $k > 2$, $AM[k] = MA[k] = AM$. Even more surprising, Goldwasser and Sipser show in [25] that for all $k \geq 1$, $IP[k] = AM$, i.e. contrary to the conjectures of [23], the $IP[k]$ hierarchy collapses, and, moreover, it collapses to AM. By the same techniques, Goldwasser and Sipser [25] also show that $AM[\text{poly}] = IP$. Thus the very restricted nature of Arthur-Merlin games does not detract in any real sense from their power as interactive proof systems, and the only new complexity classes introduced by such systems are simply MA, AM and IP. (Pedagogically, however, there is still value in having both interactive proof systems and Arthur-Merlin games around. It is easier to describe protocols in the former formalism, and easier to prove results about complexity classes in the latter.)

We conclude this section with a quick survey of what is known about MA, AM, and IP. A first observation is that $IP \subseteq PSPACE$ (and consequently, our allowing Merlin (Alice) arbitrary computing power was overkill; polynomial space is quite enough). Containment in PSPACE follows from the observation that an Arthur-Merlin game is just a special case of a “game against nature,” as defined by Papadimitriou in [30]. The latter is a variant on the alternating Turing machine (called a *stochastic Turing machine* or “STM”) in which all states are labelled as either *existential* or *random* (rather than *existential* or *universal*). An *admissible* computation for such a computation tree is a subtree obtained by deleting all but one of the subtrees rooted at each interior existential node. An STM M accepts a string x if the resulting computation tree contains an admissible subtree in which more than half of the leaves are accepting. In terms of games, this is the same as saying that there is a strategy under which the existential player has probability greater than $1/2$ of beating a random opponent (one who always chooses his next move randomly, with each of the available options

equally likely to be chosen). Papadimitriou showed that the set of languages recognizable by polynomial-time STM's is simply PSPACE. To view Arthur-Merlin games as STM's, simply identify messages from Arthur with sequences of random moves and messages from Merlin as sequences of existential moves. Under this identification, one can conclude that Arthur-Merlin games are simply polynomial-time STM's that obey more stringent requirements, in particular the following one on admissible computations: either there exists one in which $2/3$ or more of the leaves are accepting or else there is none with more than $1/3$ of the leaves accepting. Thus it follows from Papadimitriou's result that $IP \subseteq PSPACE$, and one would suspect that the containment is proper.

The next question one might ask is whether IP is contained within the polynomial hierarchy (PH). If it is, it seems unlikely that we will be able to prove this fact soon: Aiello, Goldwasser, and Hastad have constructed oracles B such that $IP^B - PH^B$ is non-empty [2]. This latter result indicates in addition that AM is likely to be a proper subclass of IP, since Babai has shown that $AM \subseteq \Pi_2^P$, a relatively low-level set in the polynomial hierarchy [4]. (MA is in $\Pi_2^P \cap \Sigma_2^P$ [4].) Another question about AM is whether it contains co-NP. We know that $NP \subseteq AM$, but the definition of AM is not symmetric, and so this does not imply that $co-NP \subseteq AM$. Boppana, Hastad, and Zachos provide strong evidence in [10] that co-NP is *not* contained in AM, given the current consensus on the non-triviality of the polynomial hierarchy. They show that if $co-NP \subseteq AM$, then the polynomial hierarchy collapses (indeed, we would have $PH \subseteq AM \subseteq \Pi_2^P$) [10].

As an important and surprising corollary, this last result provides strong evidence that GRAPH ISOMORPHISM, one of the few remaining open problems from [G&J], cannot be NP-complete. Recall that, a few pages back, we presented an interactive proof system with a fixed number of rounds for GRAPH NON-ISOMORPHISM, the complement of the GRAPH ISOMORPHISM problem (a result from [20,21]). Thus $GRAPH\ NON-ISOMORPHISM \in AM$. If GRAPH ISOMORPHISM were NP-complete, then GRAPH NON-ISOMORPHISM would be complete for co-NP, and hence, for each language L in co-NP, there would be a polynomial transformation from L to GRAPH NON-ISOMORPHISM, and we could compose this with the interactive proof system for GRAPH NON-ISOMORPHISM to obtain a proof that $L \in AM$. Thus, if GRAPH ISOMORPHISM were NP-complete, we would have $co-NP \subseteq AM$, and by the previous result, this would imply that the polynomial hierarchy collapses [10]. A slightly different proof, which shows that GRAPH ISOMORPHISM cannot even be γ -complete unless the polynomial hierarchy collapses, is presented by Schöning in [31]. (Recall that γ -completeness is one of the weakest variants on NP-completeness that still offers hope of implying intractability. The question of whether GRAPH ISOMORPHISM might be γ -complete was raised by Adleman and Manders in their original paper introducing the concept [1].)

2. PERFECT ZERO-KNOWLEDGE

It seems unlikely that Goldwasser, Micali, and Rackoff had the potential collapse of the polynomial hierarchy in mind when they introduced the concept of an interactive proof system in [23]. They were more interested in what they called the “knowledge complexity” of an interactive proof system, which they defined to be the number of bits leaked (in a technical sense) by a proof under the system. In particular, they were interested in systems that leaked no bits at all, i.e., “zero-knowledge” proofs. For instance, consider once more the interactive proof system that recognized NON-ISOMORPHIC GRAPHS. Here, assuming the given graphs are non-isomorphic, Alice never tells Bob any information that he doesn’t already know (the identity of the graph G_i from which he constructed H as a random isomorph). It is thus hard to believe that Bob actually gains any new insights from taking part in the protocol. The concept of “zero-knowledge” (in all its variant forms) is an attempt to formalize this notion.

In this section we will restrict attention to what are known as “perfect zero-knowledge proofs,” as these represent the simplest and least confusing variant on the basic ideas. We shall also introduce yet another set of names for Alice and Bob, in order to simplify and perhaps to clarify the upcoming inundation of notation. For the remainder of this column, the verifier (Bob) will be known simply as “ V ” and the prover (Alice) will be known as “ P .” (This latter is not to be confused with the non-italicized “ P ” that traditionally represents “polynomial time.” The prover, at least for now, remains unconstrained by any explicit computational bounds.)

Let (P, V) be an interactive proof system that recognizes a given language L . We start by defining what it means for (P, V) to provide “perfect zero-knowledge for verifier V .” Any enactment of the (P, V) proof system can be thought of as generating a string $v = (m_1, m_2, \dots, m_k, r)$ called a “view” or “transcript,” where the m_i are the alternating messages sent by P and V , and r is the sequence of random bits generated by V , i.e., the prefix of his random input tape that he actually read. (This must be included for technical reasons, as it may not always be deducible from V ’s messages [28].) Note that for any $x \in L$, there is a probability distribution $\langle D_{P, V(x)} \rangle$ over potential views v where, for each v , $\langle D_{P, V(x)} \rangle(v)$ is the probability that v will arise as the completed transcript of an interactive proof that $x \in L$, as generated by P and V . We say that (P, V) recognizes L with *perfect zero-knowledge for V* if there is a randomized, expected polynomial-time algorithm S (the “simulator”) that, given any x , generates strings v according to a probability distribution $\langle D_S(x) \rangle$ such that for all $x \in L$, $\langle D_S(x) \rangle = \langle D_{P, V(x)} \rangle$, i.e., $\langle D_S(x) \rangle(v) = \langle D_{P, V(x)} \rangle(v)$ for all strings v . (The definition does not care what the simulator does when $x \notin L$.)

For the GRAPH NON-ISOMORPHISM proof system, $\langle D_{P, V(x)} \rangle$ is a probability distribution on views for an x encoding some pair G_1 and G_2 of non-isomorphic graphs. For this case the view has the following format: the first

message is from V and consists of a random permutation of a randomly chosen $G_i \in \{G_1, G_2\}$. The second message is from P and consists of the index i of the graph chosen by V . The third and fourth rounds are generated in the same way, but with the index and the permutation chosen independently of the choices made in the first round. The random bits are then simply those used to generate the preceding choices. Note that it is easy to generate transcripts in this form in random polynomial time, even if we do not know whether $x \in L$. Simply have the simulator S perform V 's part of the protocol and always have P respond with the correct answer. Whenever $x \in L$, we will have $\langle D_S(x) \rangle = \langle D_{P,V}(x) \rangle$, and so the protocol, as described, qualifies as “perfect zero-knowledge for V .”

Having seen the definition and an illustration, one may still be wondering why we use the term “zero-knowledge” for this situation. In the case of the GRAPH NON-ISOMORPHISM proof system, this may seem relatively uncontroversial (assuming we do not count “conviction” as knowledge). Unfortunately for our terminological peace of mind, the technical definition of “perfect zero knowledge” allows for much subtler behavior.

For example, the following interactive proof system for GRAPH ISOMORPHISM also qualifies as providing “perfect zero-knowledge for V .” Given two isomorphic graphs G_1 and G_2 , the proof starts with P sending V a random isomorph H of G_1 . V then sends P a random index $i \in \{1, 2\}$, and P sends a description of an isomorphism from G_i to H . This whole process is performed twice, and V accepts assuming he received valid isomorphisms on each iteration. Note that V will always accept if G_1 and G_2 are isomorphic, and that the probability that a cheating prover could convince V that two non-isomorphic graphs were isomorphic is $1/4$. Thus the system recognizes the GRAPH ISOMORPHISM language.

It is easy to prove that this system provides “perfect zero-knowledge for V ” according to the technical definition. Note, however, that in this system, there are times when P *does* tell V things he presumably didn't previously know (the isomorphisms between H and G_i). (This is no problem for the simulator S that proves zero-knowledge, since on each iteration S simply picks a random i , generates a random isomorph $H = f(G_i)$, and adds “ H, i, f ” to the transcript.) Thus in this proof system, knowledge other than the conviction that G_1 and G_2 are isomorphic *is* transmitted. Note, however, that, by definition, the piece of knowledge learned could have been learned, with equal probability, had the verifier used the simulator rather than entering into the (P, V) protocol. Thus, although an individual proof might leak some information, the *proof system* (which is the object to which the property of “perfect zero-knowledge” applies) does not augment the verifier's ability to gain new knowledge.

There remains the question of whether the “conviction” that $x \in L$ provided by a zero-knowledge proof system should count as knowledge. Those who think it does prefer to call these systems “minimum knowledge” systems [16] (and count, amongst the “unavoidable” minimum amount of information

leaked, all consequences derivable in random polynomial time from the fact that $x \in L$). On the other hand, one can take the philosophical position that no fact is “known” unless one can convince a skeptical adversary of its truth (just as the old adage asserts that one doesn’t understand something until one can explain it to someone else). In this case, one can reasonably argue that, with high probability, a zero-knowledge proof that $x \in L$ does not provide the “knowledge” that $x \in L$. Even if P and V perform many iterations of the protocol, the only new information that V obtains rests in the transcripts of the proof sessions, and a suspicious third party X by definition has no way of telling whether these were generated by the (P, V) protocol, or simply by using the simulator S . If X could nonetheless determine whether $x \in L$ from these transcripts in polynomial time, then so could V , and hence the interactive proofs provided V no information he could not have derived for himself. (Readers who are still confused might wish to consult [26], where a formal, logical theory of the consequences of zero-knowledge is developed.)

There is one source of information leakage we have not yet covered, however, and that is V himself. Suppose he cheats. For instance, in the proof system for NON-ISOMORPHIC GRAPHS, instead of sending P a random isomorph of one of the G_i ’s, he could send an entirely different graph G' , and thus obtain information as to whether G' were isomorphic to either G_1 or G_2 . The definition of “perfect zero-knowledge” (unqualified by the identity of the verifier V) is designed to rule out leaks of this sort. Just as we allowed for “cheating” provers in the definition of “soundness,” we allow for “cheating” verifiers in the definition of “perfect zero-knowledge.” Moreover, we allow the cheating verifier to have an auxiliary input tape on which a “hint” may be written. (This last was not present in the original definitions of Goldwasser, Micali, and Rackoff [23], but is included in the final version [24] of that paper, and its utility was independently noted in [29,32].) It serves the specific purpose of allowing “zero-knowledge” to survive when a given protocol is iterated repeatedly or used as a subprotocol, since one role for the hint would be to record transcripts of previous sessions with P .) We say that an interactive proof system (P, V) for a language L provides *perfect zero-knowledge* if for any imposter V' there exists a simulator $S[V']$ (also with an auxiliary hint tape) such that for all $x \in L$ and all hints h of length bounded by a polynomial in $|x|$, the distributions $\langle D_{P, V'}(x, h) \rangle$ and $\langle D_{S[V']}(x, h) \rangle$ are identical.

Goldreich, Micali, and Wigderson show in [21] how to fix the leak in the GRAPH NON-ISOMORPHISM proof system so that it does provide perfect zero-knowledge. The above GRAPH ISOMORPHISM proof system, however, provides “perfect zero-knowledge” as given. The key in proving this is to replace the straightforward polynomial time simulator we used for V by an expected polynomial time simulator that deals with a potential imposter V' by using him as a restartable subroutine. Given G_1 , G_2 , and a hint h , we simulate a round of the proof system by repeating the following experiment until it

succeeds: Pick a random i , generate a random isomorph $H = f(G_i)$, place this on the interaction tape, and then restart V (rewound to the state he was in when he sent his last message). If V generates the message “ i ” (success), add “ H, i, f ” to the transcript. The probability of success is $1/2$ for each such experiment, so it is easy to argue that the expected overall running time for the simulation is polynomially bounded. For a proof that the distribution generated is appropriately indistinguishable, see [21]

Other perfect zero-knowledge proof systems are described in [32]. A natural question is how many problems in IP have such systems. In complexity class terms, what is the relationship between the class “PZK” of languages recognizable by perfect zero-knowledge proof systems and the classes previously introduced. We know by definition that $\text{PZK} \subseteq \text{IP}$. In [15] it is further shown that $\text{PZK} \subseteq \text{co-AM}$, and in [3] it is shown that $\text{PZK} \subseteq \text{AM}$. (These results also hold if we generalize PZK to include all languages recognizable by proof systems (P, V) that are merely perfect zero-knowledge for V [3,15].) Thus, the oracular results of the previous section make it unlikely that $\text{PZK} = \text{IP}$. It also appears unlikely that PZK contains NP-complete (or co-NP-complete) problems, for if it did the polynomial hierarchy would collapse (by an argument similar to the one given at the end of the previous section). We can, however, obtain zero-knowledge proofs of NP-complete problems if we accept a weaker standard for “zero-knowledge” and rely on some unproved cryptographic assumptions, and this will be one of the topics covered in the next section.

3. COMPUTATIONAL ZERO-KNOWLEDGE

How might we reasonably make the definition of “zero-knowledge” less restrictive? Given that in our model of an interactive proof the verifier V has limited computational power, one aspect of “perfect” zero-knowledge that is perhaps unnecessarily stringent is requirement that the distributions $\langle D_{P, V}(x, h) \rangle$ and $\langle D_{S[V]}(x, h) \rangle$ be identical. A less-restrictive alternative, and one that has proved useful in other contexts, is that the two ensembles of distributions parameterized by (x, h) be “indistinguishable” with respect to probabilistic polynomial-time algorithms or, more technically useful, with respect to polynomial-size circuits.

For any n -input, one-output Boolean circuit C , and any probability distribution $\langle D \rangle$ over strings in $\{0, 1\}^n$, let $p[C, \langle D \rangle]$ be the probability that C , on input randomly chosen according to $\langle D \rangle$, will output “1.” Suppose $\{\langle D_1(y) \rangle : y \in L\}$ and $\{\langle D_2(y) \rangle : y \in L\}$ are sets of probability distributions over $\{0, 1\}^*$, where all strings with non-zero probability under $\langle D_i(y) \rangle$ have length $f(|y|)$ for some fixed polynomial f . Then we say that the two ensembles are *indistinguishable (with respect to polynomial-size circuits)* if for every family $\{C_k : k \geq 1\}$ of one-output circuits, where C_k has k inputs and a number of gates bounded by a polynomial in k , we have that $|p[C_{f(|y|)}, \langle D_1(y) \rangle] - p[C_{f(|y|)}, \langle D_2(y) \rangle]| \leq |y|^{-c}$ for all constants

$c > 1$ and all sufficiently large $|y|$. (In addition to its use in defining computational zero-knowledge, this concept is also at the heart of the notions of “cryptographically secure pseudo-random number generator” [34] and, as we shall see, “secure encryption scheme” [22].)

An interactive proof system (P, V) is said to provide *computational zero-knowledge* if for each auxiliary-input imposter V' there is an auxiliary-input simulator $S[V']$ such that the two ensembles of distributions $\{\langle D_{P, V'}(x, h) \rangle : x \in L, |h| \leq p(|x|)\}$ and $\{\langle D_{S[V']}(x, h) \rangle : x \in L, |h| \leq p(|x|)\}$ are indistinguishable. (We may assume without loss of generality that the length of a view is determined by the combined length of x and h .) Note that perfect zero-knowledge implies computational zero-knowledge. Many authors simply use “zero-knowledge” to mean computational zero-knowledge, but we shall here retain the modifier for clarity, using “zero-knowledge” by itself to refer generically to all the definitions.

Given the above definition of “indistinguishable,” it might at first seem that in order to take advantage of the less-restrictive notion of “computational” zero-knowledge, one would have to be able to prove that certain computations could not be performed in (non-uniform) polynomial time. Given the current state of the art, this would seem to imply the need for unproved intractability assumptions if we are to find computational zero-knowledge proof systems for problems not known to be in PZK. Technically, this is not quite true, however. It might be possible to prove that the corresponding distributions are so close (statistically) that indistinguishability would hold even if one allowed arbitrary sized circuits. This can be formulated as the intermediate concept of “statistical zero-knowledge” (e.g. see [3,15]). Unfortunately, if our real goal is to find a computational zero-knowledge proof system that recognizes an NP-complete problem, statistical zero-knowledge is no better than perfect zero-knowledge. By analogues of the arguments about PZK mentioned above, any language recognizable by a statistical zero knowledge proof system is also in $AM \cap co-AM$ [3,15] and hence cannot be NP-complete unless the polynomial hierarchy collapses. Thus it would appear that one of those unsightly intractability assumptions will indeed be unavoidable.

Fortunately, Goldreich, Micali, and Wigderson [20,21] have shown that one need not rely on the supposed intractability of any particular computational task; it is enough to assume the existence of a generically defined “secure encryption scheme” (as introduced in [22]). For our purposes we need only to be able to encode messages from a finite set M , and I will present the relevant definitions under this assumption. A (*probabilistic*) *encryption scheme* for M is a probabilistic algorithm Π that, for any positive integer k (the *security parameter*), generates polynomial-time computable one-one *encoding functions* $E: (M \times \{0,1\}^k) \rightarrow \{0,1\}^*$. For any message $m \in M$, let $\langle D_{\Pi, m}(1^k) \rangle$ be the probability distribution over $\{0,1\}^*$ determined by first applying Π to k to obtain a random encoding function E , and then generating a random k -bit number r and computing $E(m, r)$.

(We use 1^k rather than k so that when we reapply the definition of “indistinguishability” given above, the error bound will be an inverse polynomial in k rather than one in $\log k$.) The encryption scheme Π is *secure* if for all pairs $m, m' \in M$, the distribution ensembles $\{ \langle D_{\Pi, m}(1^k) \rangle : k \geq 1 \}$ and $\{ \langle D_{\Pi, m'}(1^k) \rangle : k \geq 1 \}$ are indistinguishable. (Thus, assuming the security parameter is set high enough, not even a *non-uniform* polynomial-time adversary is likely to be able to decrypt the message, even should he be in possession of the function E .)

It can be shown that a secure encryption scheme can be derived from any “one-way permutation” (as defined by Yao in [34]). A variety of such schemes have been proposed, most based at least in part on the assumption that factoring the product of two equal-length primes is almost always difficult. One of the most popular candidates, introduced by Goldwasser and Micali [22] depends on a strengthened version of this assumption called the “quadratic residuosity assumption” (interested readers are referred to [22] for the number-theoretic details involved).

Given a “secure encryption scheme” such as this one, Goldreich, Micali, and Wigderson, in [20,21], present the following computational zero-knowledge proof system for the NP-complete GRAPH 3-COLORABILITY problem. Given $G = (U, F)$ (pardon the pseudonyms, but “ V ” and “ E ” are already taken), the prover P initially computes a 3-coloring $c: U \rightarrow \{1, 2, 3\}$ of G , keeping this to herself. Having chosen an appropriate security parameter (as a polynomial function of the size of the graph), she then uses the encryption scheme to generate a random encoding function, and sends this to the verifier. The system then iterates the following loop until the probabilities have reached the desired levels (with both parties prepared to abort the interaction if they detect cheating): The prover chooses a random permutation π of $\{1, 2, 3\}$ and, using the secure encryption scheme, sends encryptions of $\pi(c(v))$ for each $v \in V$ to the verifier. The verifier then chooses a random edge $\{u, v\}$ of G , and sends an identification of it to the prover. The prover then “reveals” the coloring of u and v , by sending the bits of $\pi(c(u))$ and $\pi(c(v))$, together with the random numbers that were used in their encryption. At this point the verifier, using E , can verify that the claimed colors did indeed give rise to the encryptions received (recall that E is one-one) and that they differ. If so, the iteration is complete and the interaction proceeds.

Note that the probability that a non-3-colorable graph will pass one iteration is no more than $1 - 1/|F|$, and so a polynomial number of successful iterations will suffice to convince the verifier that the graph is 3-colorable. The soundness and completeness of this system are thus easy to verify. The proof that it provides computational zero-knowledge is more complicated, and readers interesting in decrypting it are referred to [20,21].

Once we have shown this first NP-complete problem to have a computational zero-knowledge proof, the rest follow via the standard transformations [20,21]. Indeed, it is shown in [7,27] that all languages in IP are recognizable by a computational zero-knowledge proof systems (assuming only the existence of secure

encryption schemes). For any language L in NP, however, there is a special nicety: the prover is not required to have arbitrary power. All she needs is a short proof that $x \in L$, and then she can proceed in probabilistic polynomial time. This is certainly true for the GRAPH 3-COLORABILITY proof system above (and indeed for the perfect zero-knowledge proof system for GRAPH ISOMORPHISM in the last section.) It does not, however, directly follow for other languages L in NP, given only the definition of NP-completeness. Rather, one must use the proof of Cook's theorem, together with the standard transformation from SATISFIABILITY to GRAPH 3-COLORABILITY [G&J]. These not only yield a polynomial transformation f from L to GRAPH 3-COLORABILITY, but also show how to construct a 3-coloring of $f(x)$, given an accepting computation for x by the NDTM that recognizes L .

With the introduction of proof systems in which P can be restricted to probabilistic polynomial-time, we pass from the realm of theoretical "fun" to potential practical "profit" (and thus justify this column's title). One possible application is "zero-knowledge proofs of identity," as suggested by Feige, Fiat, and Shamir in a paper [14] that the US Army briefly attempted to classify [18]. As a civilian application, consider the task of verifying your identity to a bank's cash machine. By playing the role of the prover in a zero-knowledge interactive proof system with the cash machine, you can convince it that you know crucial knowledge available only to you, and do this in such a way that no one looking over your shoulder can figure out how to impersonate you, not even someone who has illicitly obtained long-term "superuser" privileges on the computer that controls the cash machine. (This would certainly be an advantage over the current scheme whereby one types in a "secret" 4-digit number, although it is unlikely to be as fast.)

Another application would be as a subprotocol in a more complicated system where the various players don't necessarily trust each other to obey the rules and may have secrets to protect. The fact that all the messages sent to date by player A are consistent with the rules of the protocol will typically be a property in NP, with a short proof consisting of the random bits the player has generated so far, together possibly with some initial secret. This short proof would naturally be known to player A , and so by the result above (and assuming that secure encryption schemes exist), there exists a computational zero-knowledge proof system whereby he can convince his teammates/opponents that he has not been cheating, without revealing anything else. (For more detailed implementations, see [12,20,21,27].)

Restricting the prover to probabilistic polynomial time provides additional opportunities, as it opens up the way for what might be called "weakly sound" proof systems, where soundness depends on the polynomial-time bounds on the prover (as well perhaps as some more of those unproved cryptographic assumptions). For instance, it is shown in [11] that, on the assumption that factoring is appropriately hard, every L in NP has a weakly sound *perfect* zero-knowledge

proof system (a possibility that was ruled out above for proof systems that are truly sound, assuming the polynomial hierarchy does not collapse). Another way to change the basic model so that we can get perfect zero-knowledge proofs for NP is to introduce a second prover. Assuming certain limitations on the communication between the two provers, it is shown in [8] that all languages in NP have (truly sound) perfect zero-knowledge proof systems in this latter model.

As a final item in this survey of the susceptibility of NP-complete problems to zero-knowledge proof systems, let me mention a recent paper that is currently discomfiting many of us who thought we understood why zero-knowledge proof systems work for hard problems. In all the examples given above, the key to the completeness and soundness of the systems was that the prover did not have access to the verifier's random bits. When Goldwasser and Sipser showed that $IP = AM[\text{poly}]$ in [22], they allowed us to conclude that interactive proof systems lost no power if they lost that privacy. The privacy of V 's random bits still seemed necessary, however, if the proof system was to be zero-knowledge, as did the inclusion of non-trivial interaction between P and V . In [9], Blum, Feldman, and Micali exhibit computational zero-knowledge proof systems that recognize NP-complete languages (in particular, GRAPH 3-COLORABILITY again) and in which all the verifier's random bits are made public at the beginning of the protocol, and in which only one other communication is made, this being from the prover to the verifier. There is still *some* secrecy, however (the prover gets to keep her own random bits secret), and the verifier's bits must be *guaranteed* to be random. Moreover, as presented in the paper (and in [13]), the result depends on the quadratic residuosity assumption, and hence on a more specialized intractability assumption than the mere existence of a secure encoding scheme. Nevertheless, it now appears that, given randomness, secrecy is not nearly so important as we thought, even when one's goal is to keep as many secrets as possible.

4. GENUS IS NP-COMPLETE

Of the twelve problems on the original open problem list in [G&J], one of the few to have stood the test of time (until now) was the "GENUS" problem: given a graph $G = (V, E)$ and an integer k , can G be embedded on a surface of genus k (one with at most k "handles") in such a way that no two edges cross one another? This problem was known to be solvable in polynomial time if k was fixed, although the algorithms had running times that were exponential in k [33].

Carsten Thomassen has now shown what most of us suspected: if k is not fixed, the problem becomes NP-complete. His proof involves several intricate arguments, but its overall structure is that of a straightforward "local replacement" transformation from VERTEX COVER to GENUS. Key to the proof is

a new technical lemma providing a simple way to modify any graph so as to increase its genus by exactly one (without knowing the original value). Details can be found in the forthcoming paper [33].

For those who wish to keep score, this new result means that of the original twelve problems in [G&J], four have been shown to be NP-complete, four are polynomial-time solvable, and four remain open (GRAPH ISOMORPHISM, COMPOSITE NUMBER, 3-PROCESSOR SCHEDULING, and MINIMUM LENGTH TRIANGULATION).

REFERENCES

1. L. ADLEMAN AND K. MANDERS, Reducibility, randomness, and intractability, in "Proceedings 9th Ann. ACM Symp. on Theory of Computing," pp. 151-163, Association for Computing Machinery, New York, 1977.
2. W. AIELLO, S. GOLDWASSER, AND J. HASTAD, On the power of interaction, in "Proceedings 27th Ann. Symp. on Foundations of Computer Science," pp. 368-379, IEEE Computer Society, Los Angeles, 1986.
3. W. AIELLO AND J. HASTAD, Perfect zero-knowledge languages can be recognized in two rounds, in "Proceedings 28th Ann. Symp. on Foundations of Computer Science," pp. 439-448, IEEE Computer Society, Los Angeles, 1987.
4. L. BABAI, Trading group theory for randomness, in "Proceedings 17th Ann. ACM Symp. on Theory of Computing," pp. 421-429, Association for Computing Machinery, New York, 1985.
5. L. BABAI AND E. SZEMERÉDI, On the complexity of matrix group problems, in "Proceedings 25th Ann. Symp. on Foundations of Computer Science," pp. 229-240, IEEE Computer Society, Los Angeles, 1984.
6. C. H. BENNETT AND J. GILL, Relative to a random oracle A , $P^A \neq NP^A \neq co-NP^A$ with probability 1, *SIAM J. Comput.* **10** (1981), 96-113.
7. M. BEN-OR, O. GOLDREICH, S. GOLDWASSER, J. HASTAD, J. KILIAN, S. MICALI, AND P. ROGAWAY, Everything provable is provable in zero-knowledge, in "Advances in Cryptology: CRYPTO '88 Proceedings," Lecture Notes in Computer Science, Springer, Berlin, 1988 (to appear).
8. M. BEN-OR, S. GOLDWASSER, J. KILIAN, AND A. WIGDERSON, Multi-prover interactive proofs: How to remove intractability assumptions, in "Proceedings 20th Ann. ACM Symp. on Theory of Computing," pp. 113-131, Association for Computing Machinery, New York, 1988.
9. M. BLUM, S. MICALI, AND P. FELDMAN, Non-interactive zero-knowledge and its applications, in "Proceedings 20th Ann. ACM Symp. on Theory of Computing," pp. 103-112, Association for Computing Machinery, New York, 1988.
10. R. B. BOPPANA, J. HASTAD, AND S. ZACHOS, Does co-NP have short interactive proofs?, *Inform. Process. Lett.* **25** (1987), 127-133.
11. G. BRASSARD AND C. CREPEAU, Non-transitive transfer of confidence: A perfect zero-knowledge interactive protocol for SAT and beyond, in "Proceedings 27th Ann. Symp. on Foundations of Computer Science," pp. 188-195, IEEE Computer Society, Los Angeles, 1986.
12. D. CHAUM, I. B. DAMGARD, AND J. VAN DE GRAAF, Multiparty computations ensuring privacy of each party's input and correctness of the result, "Advances in Cryptology: CRYPTO '88 Proceedings" (C. Pomerance, Ed.), pp. 86-119, Lecture Notes in Computer Science, Vol. 293, Springer, Berlin, 1987.
13. A. DE SANTIS, S. MICALI, AND G. PERSIANO, Non-interactive zero-knowledge proof systems, "Advances in Cryptology: CRYPTO '88 Proceedings" (C. Pomerance, Ed.), pp. 52-72, Lecture Notes in Computer Science, Vol. 293, Springer, Berlin, 1987.

14. U. FEIGE, A. FIAT, AND A. SHAMIR, Zero-knowledge proofs of identity, in "Proceedings 19th 210-217 Ann. ACM Symp. on Theory of Computing," Association for Computing Machinery, New York, 1987.
15. L. FORTNOW, The complexity of perfect zero-knowledge, in "Proceedings 19th Ann. ACM Symp. on Theory of Computing," pp. 204-209, Association for Computing Machinery, New York, 1987.
16. Z. GALIL, S. HABER, AND M. YUNG, A private interactive test of a Boolean predicate and minimum-knowledge public-key cryptosystems, in "Proceedings 26th Ann. Symp. on Foundations of Computer Science," pp. 360-371, IEEE Computer Society, Los Angeles, 1985.
17. J. GILL, Computational complexity of probabilistic Turing machines, *SIAM J. Comput.* **6** (1977), 675-695.
18. J. GLEICK, Brief U.S. suppression of proof stirs anger, *New York Times* (February 17, 1987), C3.
19. O. GOLDREICH, Y. MANSOUR, AND M. SIPSER, Interactive proof systems: Provers that never fail and random selection, in "Proceedings 28th Ann. Symp. on Foundations of Computer Science," pp. 449-461, IEEE Computer Society, Los Angeles, 1987.
20. O. GOLDREICH, S. MICALI, AND A. WIGDERSON, Proofs that yield nothing but their validity and a methodology of cryptographic protocol design, in "Proceedings 27th Ann. Symp. on Foundations of Computer Science," pp. 174-187, IEEE Computer Society, Los Angeles, 1986.
21. O. GOLDREICH, S. MICALI, AND A. WIGDERSON, "Proofs that yield nothing but their validity or All languages in NP have zero-knowledge proofs," Report No. 498, Computer Science Department, Technion, Haifa, Israel, February 1988.
22. S. GOLDWASSER AND S. MICALI, Probabilistic encryption, *J. Comput. System Sci.* **28** (1984), 270-299.
23. S. GOLDWASSER, S. MICALI, AND C. RACKOFF, The knowledge complexity of interactive proof-systems, in "Proceedings 17th Ann. ACM Symp. on Theory of Computing," pp. 291-304, Association for Computing Machinery, New York, 1985.
24. S. GOLDWASSER, S. MICALI, AND C. RACKOFF, The knowledge complexity of interactive proof-systems, *SIAM J. Comput.*, to appear.
25. S. GOLDWASSER AND M. SIPSER, Private coins versus public coins in interactive proof systems, in "Proceedings 18th Ann. ACM Symp. on Theory of Computing," pp. 59-68, Association for Computing Machinery, New York, 1986.
26. J. Y. HALPERN, Y. MOSES, AND M. R. TUTTLE, A knowledge-based analysis of zero knowledge, in "Proceedings 20th Ann. ACM Symp. on Theory of Computing," pp. 132-147, Association for Computing Machinery, New York, 1988.
27. R. IMPAGLIAZZO AND M. YUNG, Direct minimum-knowledge computations, "Advances in Cryptology: CRYPTO '88 Proceedings" (C. Pomerance, Ed.), pp. 40-51, Lecture Notes in Computer Science, Vol. 293, Springer, Berlin, 1987.
28. S. MICALI, private communication (1988).
29. Y. OREN, On the cunning power of cheating adversaries: Some observations about zero-knowledge proofs, in "Proceedings 28th Ann. Symp. on Foundations of Computer Science," pp. 462-471, IEEE Computer Society, Los Angeles, 1987.
30. C. H. PAPADIMITRIOU, Games against nature, in "Proceedings 24th Ann. Symp. on Foundations of Computer Science," pp. 446-450, IEEE Computer Society, Los Angeles, 1983.
31. U. SCHÖNING, Graph isomorphism is in the low hierarchy, manuscript (1986).
32. M. TOMPA AND H. WOLL, Random self-reducibility and zero-knowledge interactive proofs of possession of information, in "Proceedings 28th Ann. Symp. on Foundations of Computer Science," pp. 472-482, IEEE Computer Society, Los Angeles, 1987.
33. C. THOMASSEN, The graph genus problem in NP-complete, *J. Algorithms*, to appear.
34. A. C. YAO, Theory and applications of trapdoor functions, in "Proceedings 23rd Ann. Symp. on Foundations of Computer Science," pp. 80-91, IEEE Computer Society, Los Angeles, 1982.