Contents

Probabilistic Proof Systems – Part I 1

Lecture 1. Interactive Proofs 1
1.1. Definitions 2
1.2. GRAPH NONISOMORPHISM 3
1.3. **co-NP** and more 5
1.4. Additional Topics 9
1.5. Exercises 11

Lecture 2. Zero-Knowledge Proofs 15
2.1. Definition 15
2.2. Zero-knowledge Proofs for **NP** 16
2.3. Additional Topics 22
2.4. Exercises 25

Suggestions for Further Reading 27

Bibliography 29

IAS/Park City Mathematics Series
Volume 10, 2000

Probabilistic Proof Systems — Part I

Salil Vadhan

LECTURE 1
Interactive Proofs

The notion of a *proof* is central to mathematics and computer science, and hence has been the subject of much investigation in both fields. Indeed, from previous lectures in this volume, the reader should already be aware of the intimate connection between traditional mathematical proofs and the fundamental questions of complexity theory (*e.g.*, $\mathbf{P} \stackrel{?}{=} \mathbf{NP}$ and $\mathbf{NP} \stackrel{?}{=} \mathbf{co}\text{-}\mathbf{NP}$). In this lecture series (and the subsequent one by Madhu Sudan), we will examine several *nontraditional* notions of proof which have been at the center of some very exciting developments in complexity theory.

Recall that proofs are given their meaning by specifying a procedure for verifying them. To formalize this, both assertions and proofs are written as strings over some finite alphabet, and a language L is used to identify the strings representing "true assertions." A *classical proof system* for L is given by a verification algorithm V with the following two properties:

1. (Completeness) True assertions have proofs. That is, if $x \in L$, then there exists *proof* such that $V(x, \mathit{proof}) = \texttt{accept}$.
2. (Soundness) False assertions have no proofs. That is, if $x \notin L$, then for all proof^*, $V(x, \mathit{proof}^*) = \texttt{reject}$.
3. (Efficiency) $V(x, \mathit{proof})$ runs in time $\mathrm{poly}(|x|)$.

Clearly, completeness and soundness are central to our intuitive notion of proof. Some form of efficiency is also important, for if one could decide whether the assertion is true in less time than it takes to verify the proof, then the proof loses its usefulness. Recall that **NP** is the class of languages having classical proof systems as defined above.

---

[1]Division of Engineering and Applied Sciences, Harvard University, Cambridge, MA.
E-mail address: `salil@deas.harvard.edu`.
The author is supported by an NSF Mathematical Sciences Postdoctoral Research Fellowship. Thanks to Jirka Hanika for assistance in preparing these lecture notes.

In these lectures, we will consider augmenting the above notion with two new ingredients (as proposed in [**GMR89, BM88**]). The first is *randomization*; that is, we will allow the verification procedure to toss coins and accept or reject incorrectly with some small probability. While this is a substantial deviation from the classical viewpoint whereby proofs establish the truth of an assertion with *certainty*, it is natural given the wide acceptance of randomized computations as reasonable substitutes for deterministic ones. The second new ingredient is *interaction*. Classically, proofs are viewed as static objects that are written and fixed, before being examined in their entirety by the verification procedure. Instead, we will allow the verifier to interact with a dynamic, all-powerful "prover" who will try to convince the verifier of the validity of the assertion at hand.

Since the classical notion of proof seems to be adequate, the reader may wonder what we gain by augmenting proof systems in these ways. Most directly, we obtain a more general notion of "efficiently verifiable proofs" which, in addition to having possible philosophical value, provides efficient proofs for more assertions than classical proofs do (as we shall see in Section 1.3). The new notions are also very useful for statements that do possess classical proofs. For example, they can yield dramatic efficiency savings in verification (as we will see in the PCP Theorem presented in Madhu Sudan's lectures). The new notions also enable us to to define and achieve properties that are meaningless (or trivial) for classical proofs. For example, in Lecture 2 we will construct *zero-knowledge* proofs, which are proofs that reveal nothing other than the validity of the assertion being proven! We also obtain new insight into classical proofs and complexity classes by characterizing them in terms of the new types of proof systems. Finally, the new types of proof systems have applications to other topics in computer science: the probabilistically checkable proofs of Madhu Sudan's lectures yield insight into the approximability of optimization problems (cf., the lectures of Sanjeev Arora in this volume) and the zero-knowledge proofs of Lecture 2 have wide applicability in cryptographic protocols (indeed, this was one of the main motivations of [**GMR89**]).

1.1. Definitions

Basic Notation: Let A be a probabilistic algorithm. $A(x;r)$ denotes the output of A when fed input x and coin tosses r. $A(x)$ denotes the distribution of $A(x;r)$ when r is chosen uniformly at random. We say that A runs in time $t(n)$ if for all x of length n, $A(x;r)$ halts within $t(n)$ steps with probability 1 over the choice of r.

As suggested above, we will obtain a new type of proof system by replacing classical (**NP**) proofs with a "prover" that "interacts" with a probabilistic "verifier". In order to make this precise, we must first formalize the notion of an interactive protocol between two parties A and B. We do this by viewing each party as a function, taking the history of the protocol (all the messages previously exchanged) and the party's random coins, to the party's next message. Either party can decide to halt the interaction (possibly accepting or rejecting), at which point the other party is given an opportunity to compute one more message.

Definition 1.1 (interactive protocols). *An* interactive protocol (A, B) *is any pair of functions from strings to strings. The* interaction *between A and B on common input x is the following random process (denoted $(A, B)(x)$):*

1. *Uniformly choose random coin tosses r_A and r_B for A and B, respectively.*

2. *Repeat the following for $i = 1, 2, \ldots$:*
   (a) *If i is odd, let $m_i = A(x, m_1, \ldots, m_{i-1}; r_A)$.*
   (b) *If i is even, let $m_i = B(x, m_1, \ldots, m_{i-1}; r_B)$.*
   (c) *If $m_{i-1} \in \{\texttt{accept}, \texttt{reject}, \texttt{halt}\}$, then exit loop.*

If the last message computed by A is `accept` *(resp.,* `reject`*), we say that A* accepts *(resp.,* rejects*), and similarly for B. We call such a protocol* polynomially bounded *if there is a polynomial $p(\cdot)$ such that, on common input x, at most $p(|x|)$ messages are exchanged, and each is of length at most $p(|x|)$ (with probability 1 over the choice of r_A and r_B).*

Originally, interactive protocols were defined in terms "interactive Turing machines," but that approach is too tied to a particular model of computation for our tastes.

Now interactive proofs can be defined as a type of interactive protocol between a prover (with no computational limitations) and a polynomial-time verifier. The completeness and soundness conditions of classical proofs are replaced with probabilistic ones which guarantee that the verifier gains statistical confidence that the assertion being proven is true.

Definition 1.2 (interactive proofs — **IP** [**GMR89, BM88**]). *An interactive protocol (P, V) is said to be an* interactive proof system *for a language L if the following conditions hold:*

1. *(Efficiency) (P, V) is polynomially bounded and V is polynomial-time computable.*
2. *(Completeness) If $x \in L$, then V accepts with probability at least $2/3$ in $(P, V)(x)$.*
3. *(Soundness) If $x \notin L$, then for any P^*, V accepts with probability at most $1/3$ in $(P^*, V)(x)$.*

IP *is class of languages possessing interactive proofs.*

We now make some basic observations about the above definition.

- The acceptance probabilities of $2/3$ and $1/3$ allowed in the above definition are arbitrary, and can be replaced with any pair of constants $1 > \alpha > \beta > 0$. Indeed, the error probability of any such proof system can be made exponentially small by taking polynomially many repetitions and having the verifier accept according to majority/threshold rule.
- Interactive proofs do indeed generalize classical proofs, because the prover can simply send the verifier a classical proof, which the verifier then checks. Thus, $\mathbf{NP} \subset \mathbf{IP}$. The main question we will address in this lecture is whether **IP** is strictly bigger than **NP**, and by how much. It is left as an exercise to prove the upper bound $\mathbf{IP} \subset \mathbf{PSPACE}$.
- The verifier's randomness is essential in interactive proofs: **IP** with deterministic verifiers collapses to **NP** (exercise). On the other hand, restricting to a deterministic prover causes no loss of generality (exercise).

1.2. Graph Nonisomorphism

Our first hint that interactive proofs are strictly more powerful than classical ones will come from an elegant proof system for Graph Nonisomorphism.

Definition 1.3. *If $G = ([n], E)$ is an undirected graph*[1] *and π is a permutation on $[n]$, then $\pi(G)$ denotes the graph obtained by permuting the vertices of G according to π. That is, $\pi(G) = ([n], E')$, where $E' = \{(\pi(u), \pi(v)) : (u, v) \in E\}$. If G and H are graphs on n vertices, and there exists a π such that $\pi(G) = H$, we say that G and H are* isomorphic *and write $G \cong H$. π is called an* isomorphism *between G and H, and H is said to be an* isomorphic copy *of G.* Graph Isomorphism *is the language* $\mathrm{GI} = \{(G, H) : G \cong H\}$. Graph Nonisomorphism *(GNI) is the complement of* GI.

It is easy to see that Graph Isomorphism is in **NP**: an easily verifiable proof that two graphs are isomorphic is an isomorphism between them. In contrast, no classical proofs are known for Graph Nonisomorphism. Nevertheless, as we shall see, Graph Nonisomorphism does possess a very efficient *interactive* proof:[2]

Theorem 1.4 ([**GMW91**]). Graph Nonisomorphism *is in* **IP**.

The interactive proof is based on two observations. First, if two graphs are nonisomorphic, then their sets of isomorphic copies are disjoint. Second, if two graphs are isomorphic, then a random isomorphic copy of one graph is indistinguishable from a random isomorphic copy of the other. Thus, the interactive proof, given in Protocol 1.5, tests whether the prover can distinguish between random isomorphic copies of the two graphs.

Protocol 1.5: Interactive proof (P, V) for Graph Nonisomorphism

Input: Graphs $G_0 = ([n], E_0)$ and $G_1 = ([n], E_1)$

1. V: Uniformly select $b \in \{0, 1\}$. Uniformly select a permutation π on $[n]$. Let $H = \pi(G_b)$. Send H to P.
2. P: If $G_0 \cong H$, let $c = 0$. Else let $c = 1$. Send c to V.
3. V: If $c = b$, accept. Otherwise, reject.

We now verify that this protocol meets the definition of an interactive proof.

Proof of Theorem 1.4 (sketch). If G_0 and G_1 are nonisomorphic, then $G_0 \cong H$ if and only if $b = 0$. So the prover strategy specified above will make the verifier accept with probability 1. Thus, completeness is satisfied.

On the other hand, if G_0 and G_1 are isomorphic, then H has the same distribution when $b = 0$ as it does when $b = 1$ (exercise). Thus, b is independent of H and the prover has at most probability at most $1/2$ of guessing b correctly *no matter what strategy it follows.* This shows that the protocol is sound. □

A few remarks about the proof system are in order. The first is it achieves an acceptance probability of 1 in the completeness condition; this attractive property is often referred to as *perfect completeness.* Second, the proof system is very communication efficient: only two messages are exchanged and the prover sends only

---

[1] To avoid notational confusion with the verifier strategy V, all of our graphs will have vertex set $[n] \stackrel{\mathrm{def}}{=} \{1, \ldots, n\}$ for some $n \in \mathbb{N}$.

[2] There has been some recent evidence that Graph Nonisomorphism is in **NP**, in fact based on the existence of an efficient interactive proof for Graph Nonisomorphism [**AK97, KvM99, MV99**].

one bit to the verifier (more generally, k bits to achieve soundness probability $1/2^k$). Finally, note that it is crucial for soundness that the verifier's random coin flips are kept "private." If the bit b is made public and revealed to the prover, soundness will no longer hold. Surprisingly, every *private-coin* interactive proof (like the one above) can be transformed into a *public-coin* one; that is, one in which the verifier's coin flips are completely visible to the prover [**GS89**].

1.3. co-NP and more

In the previous section, we saw an interactive proof for a problem not known to have efficient classical proofs, giving the first evidence that **IP** is strictly larger than **NP**. In this section, we shall obtain much stronger evidence:

Theorem 1.6 ([**LFKN92**]). **co-NP** $\subset$ **IP**.

It is widely believed that **NP** $\neq$ **co-NP** (cf., the lectures of Paul Beame in this volume), so this strongly suggests that interactive proofs are more powerful than classical ones.

1.3.1. A First Attempt

By the **NP**-completeness of SATISFIABILITY, proving that **co-NP** $\subset$ **IP** is equivalent to giving an interactive proof for UNSATISFIABILITY. So let us consider how one may try to prove that a formula φ is unsatisfiable. Actually, it will be useful to consider how to prove that a formula φ has exactly k satisfying assignments for any k. That is, we want to give an interactive proof for EXACT #SAT, the language

$$\text{E\#SAT} \stackrel{\text{def}}{=} \{(\varphi, k) : \varphi \text{ has exactly } k \text{ satisfying assignments}\}$$

Observation. *A formula $\varphi(x_1, \ldots, x_n)$ has exactly k satisfying assignments iff there exist k_0, k_1 such that*

1. $k_0 + k_1 = k$,
2. $\varphi_0(x_2, \ldots, x_n) \stackrel{\text{def}}{=} \varphi(0, x_2, \ldots, x_n)$ *has exactly k_0 satisfying assignments, and*
3. $\varphi_1(x_2, \ldots, x_n) \stackrel{\text{def}}{=} \varphi(1, x_2, \ldots, x_n)$ *has exactly k_1 satisfying assignments.*

This observation suggests a first idea for proving that φ has exactly k satisfying assignments: First, the prover sends the verifier k_0 and k_1. Second, the verifier checks that $k_0 + k_1 = k$, and randomly selects a value $b \in \{0, 1\}$ for the first variable. Then the prover recursively proves to the verifier (using the same protocol) that φ_b has exactly k_b satisfying assignments. (At the bottom of the recursion when the formula has no variables, the verifier simply checks that evaluates to 0 or 1 according to whether the prover has claimed that it has 0 or 1 satisfying assignments, respectively.)

When φ has exactly k satisfying assignments, the verifier will accept with probability 1 in this protocol. Conversely, when φ does not have exactly k satisfying assignments, one of the conditions in the observation must fail to hold, so there is a nonzero probability that the prover will continue to have a false statement to prove (unless $k_0 + k_1 \neq k$, in which case the verifier will reject immediately). Continuing this argument inductively, we conclude that the verifier has a nonzero probability of rejecting overall. However, it is not an interactive proof because, in the soundness case, the verifier may accept with probability $1 - 2^{-n}$, which is not sufficiently bounded away from 1. This is because, for each variable of the formula, the verifier may have only probability $1/2$ of setting the variable in a way that leaves the prover with something false to prove.

1.3.2. Arithmetization

Intuitively, the problem described above comes from the fact that every variable of the formula has only two possible values and we can only guarantee that at least one of these values will reflect the falsity of the assertion that the prover is trying to prove. An idea for solving this is to allow the variables to take values in a larger set $\mathbb{F}$ ($\supset \{0,1\}$), and extend the formula $\varphi : \{0,1\} \to \{0,1\}$ to a more "robust" function $\tilde{\varphi} : \mathbb{F}^n \to \mathbb{F}$ so that "most" evaluation points will reflect inconsistencies.

We will do this extension via powerful technique known as *arithmetization*. We will take $\mathbb{F}$ to be a sufficiently large finite field and show how to extend φ to a (multivariate) low-degree polynomial over $\mathbb{F}$. The robustness properties we desire will be based on the fact that two distinct low-degree polynomials cannot agree in many places.

We recursively define a mapping $\varphi \mapsto \tilde{\varphi}$ from Boolean formulas in variables $x_1, \ldots, x_n$ to polynomials over $\mathbb{F}$ in variables $x_1, \ldots, x_n$:

$$\begin{aligned} \tilde{x_i} &= x_i \\ \widetilde{\neg\varphi} &= 1 - \tilde{\varphi} \\ \widetilde{\varphi \wedge \psi} &= \tilde{\varphi} \cdot \tilde{\psi} \end{aligned}$$

(Without loss of generality, we restrict our attention to formulas over the complete basis $\neg$ and $\wedge$.)

The following are easily verified by induction:

1. $\tilde{\varphi}|_{\{0,1\}^n} = \varphi$.
2. The (total) degree of the polynomial $\tilde{\varphi}$ is at most $d = |\varphi|$.

Proving that φ has exactly k satisfying assignments is equivalent to proving

$$k = \sum_{x_1 \in \{0,1\}} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \tilde{\varphi}(x_1, \ldots, x_n) \tag{1.7}$$

(provided that the characteristic of $\mathbb{F}$ is greater than 2^n, which can be guaranteed by choosing $\mathbb{F} = \mathbb{Z}/qZ$ for a prime $q > 2^n$). The protocol for proving Equation (1.7) will proceed analogously to the first attempt above, generalized to this setting where the variables can take values in $\mathbb{F}$. The prover will send the verifier the values

$$k_\alpha \stackrel{\text{def}}{=} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \tilde{\varphi}(\alpha, x_2, \ldots, x_n) \tag{1.8}$$

for every $\alpha \in \mathbb{F}$ (rather than just k_0 and k_1 as before). As before, the verifier will check that $k_0 + k_1 = k$, and then choose a random $\alpha \in \mathbb{F}$ on which the prover should recursively prove that Equation (1.8) holds. The key observation which makes this work is that the k_α's can all be specified by a degree d polynomial p satisfying $p(\alpha) = k_\alpha \;\; \forall \alpha$ (because $\tilde{\varphi}$ is of degree d). This helps in two ways. First, it allows all the values $\{k_\alpha\}$ to be specified succinctly by the prover by giving the $d+1$ coefficients of p. (The entire list given explicitly would be of size $|\mathbb{F}| > 2^n$, which is too large). Second, it guarantees that if the prover sends a wrong value for a single k_α, then the prover must send a wrong value for *most* k_α's.

1.3.3. The Proof System

Formalizing the above ideas, we obtain Protocol 1.9.

Protocol 1.9: Interactive Proof for E#SAT

Input: A formula $\varphi(x_1, \ldots, x_n)$ and an integer k

1. P, V: Let $d = |\varphi|$, and let $\mathbb{F}$ be a finite field of characteristic greater than 2^d ($\geq 2^n$), and let $\tilde{\varphi}(x_1, \ldots, x_n)$ be the arithmetization of φ (over $\mathbb{F}$).
2. P: Compute the degree d polynomial
$$p_1(x) \stackrel{\text{def}}{=} \sum_{x_2 \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \tilde{\varphi}(x, x_2, \ldots, x_n),$$
and send p_1 to V.
3. V: Check that $p_1(0) + p_1(1) = k$ (and reject immediately if not).
4. V: Choose α_1 uniformly from $\mathbb{F}$ and send α_1 to P.
5. P, V: From $i = 2$ to n, do the following:
   (a) P: Compute the degree d polynomial
   $$p_i(x) \stackrel{\text{def}}{=} \sum_{x_{i+1} \in \{0,1\}} \cdots \sum_{x_n \in \{0,1\}} \tilde{\varphi}(\alpha_1, \ldots, \alpha_{i-1}, x, x_{i+1}, \ldots, x_n),$$
   and send p_i to V.
   (b) V: Check that $p_i(0) + p_i(1) = p_{i-1}(\alpha_{i-1})$ (and reject immediately if not).
   (c) V: Choose α_i uniformly from $\mathbb{F}$ and send α_i to P.
6. V: Accept if $p_n(\alpha_n) = \tilde{\varphi}(\alpha_1, \ldots, \alpha_n)$.

Proposition 1.10. *Protocol 1.9 is an interactive proof system for* Exact #SAT.

Proof. Efficiency can be verified by inspection. Also by inspection, we see that if φ has exactly k satisfying assignments and the prover computes all the p_i's according to the specified protocol, then all the verifier's checks will pass. That is, $p_1(0) + p_1(1) = k$, $p_i(0) + p_i(1) = p_{i-1}(\alpha_{i-1})$ for all $i > 1$, and $p_n(\alpha_n) = \tilde{\varphi}(\alpha_1, \ldots, \alpha_n)$.

Thus, we need only prove soundness. We will argue that if φ does not have k satisfying assignments, then, no matter what strategy P^* the prover follows, the verifier will accept with probability at most $nd/|\mathbb{F}| < d^2/2^d < 1/3$ (for sufficiently large $d = |\varphi|$).

Let $p_1(x), \ldots, p_n(x)$ denote the polynomials computed correctly (as prescribed by Protocol 1.9), and let $p_1^*(x), \ldots, p_n^*(x)$ denote the polynomials sent by P^*. Note that $p_1(0) + p_1(1)$ is exactly the number of satisfying assignments of φ. Thus, if φ does not have exactly k satisfying assignments, then no matter what p_1^* the prover sends, either (a) $p_1^*(0) + p_1^*(1) \neq k$, or (b) $p_1^* \neq p_1$. If (a) holds, then the verifier will reject immediately. If (b) holds, then with high probability ($\geq 1 - d/|F|$) $p_1^*(\alpha_1) \neq p_1(\alpha_1)$ (because p_1^* and p_1 are distinct degree d polynomials, and hence agree on at most d points). Thus, after the first variable is set, the prover will be left with a false assertion to prove with high probability (rather than probability 1/2), as desired.

Later rounds are analyzed in a similar fashion. Assume that

$$p_{i-1}^*(\alpha_{i-1}) \neq p_{i-1}(\alpha_{i-1}) = p_i(0) + p_i(1).$$

Then no matter what p_i^* the prover sends, it must be the case that either (a) $p_i^*(0) + p_i^*(1) \neq p_{i-1}^*(\alpha_{i-1})$, or (b) $p_i^* \neq p_i$. As before, if (a) holds the verifier will reject immediately, and if (b) holds, then $p_i^*(\alpha_i) \neq p_i(\alpha_i)$ with probability at least $1 - d/|\mathbb{F}|$.

By a union bound, it follows that, with probability at least $1 - nd/|\mathbb{F}|$, the verifier rejects or $p_n^*(\alpha_n) \neq p_n(\alpha_n) = \tilde{\varphi}(\alpha_1, \ldots, \alpha_n)$. Since the verifier will also reject in the latter case, soundness is established. □

1.3.4. A Full Characterization

co-NP $\subset$ **IP** (Thm. 1.6) follows from Proposition 1.10 because UNSATISFIABILITY reduces to EXACT #SAT via the map $\varphi \mapsto (\varphi, 0)$. In fact, it even follows that $\mathbf{P}^{\#\mathbf{P}} \subset \mathbf{IP}$. With some additional ideas, we obtain a complete characterization of the power of interactive proofs.

Theorem 1.11 ([**Sha92**]). **IP = PSPACE**

Proof sketch. Recall that a complete problem for **PSPACE** is QUANTIFIED BOOLEAN FORMULAE (QBF), *i.e.,* the language of true assertions of the form

$$\forall x_1 \exists x_2 \forall x_3 \cdots \exists x_n \varphi(x_1, \ldots, x_n),$$

where φ is a Boolean formula. Let's attempt to directly extend the ideas of Protocol 1.9 to this problem. That is, extend the arithmetization to formulas with quantifiers, and construct a protocol which eliminates one variable/quantifier at a time (with the verifier choosing random values in some field). Let $\varphi(x_1, \ldots, x_i)$ be a partially quantified formula with free (*i.e.,* unquantified) variables $x_1, \ldots, x_i$ (and "bound" variables $x_{i+1}, \ldots, x_n$). We define its arithmetization $\tilde{\varphi}(x_1, \ldots, x_i)$ as follows. If φ has no quantifiers (*i.e.,* $i = n$), then $\tilde{\varphi}$ is defined just as in Section 1.3.2. If $\varphi = \forall x_{i+1} \psi(x_1, \ldots, x_{i+1})$ then

$$\tilde{\varphi}(x_1, \ldots, x_i) = \tilde{\psi}(x_1, \ldots, x_i, 0) \cdot \tilde{\psi}(x_1, \ldots, x_i, 1) \tag{1.12}$$

If $\varphi = \exists x_{i+1} \psi(x_1, \ldots, x_{i+1})$ then

$$\tilde{\varphi}(x_1, \ldots, x_i) = 1 - \Big(1 - \tilde{\psi}(x_1, \ldots, x_i, 0)\Big) \cdot \Big(1 - \tilde{\psi}(x_1, \ldots, x_i, 1)\Big) \tag{1.13}$$

This arithmetization maintains the property that the arithmetized formulas agree with original formulas whenever the free variables are assigned values from $\{0, 1\}$. In particular, proving that a fully quantified Boolean formula is in QBF is equivalent to proving that its arithmetization is the constant polynomial 1.

The problem with this new arithmetization is that the degrees blow up, squaring with every quantifier. The result is the polynomials the prover would have to send in a protocol like Protocol 1.9 would be of exponentially large degree, and the proof system will fail to satisfy the efficiency requirement. The solution is to introduce operations that reduce the degree but have no effect on boolean values. Suppose $f(x_1, \ldots, x_i)$ is a polynomial and, for some $j \in \{1 \ldots, i\}$, consider the polynomial

$$\begin{aligned} f'(x_1, \ldots, x_i) \quad &= \quad x_j \cdot f(x_1, \ldots, x_{j-1}, 1, x_{j+1}, \ldots, x_n) + \\ &\quad\quad (1 - x_j) \cdot f(x_1, \ldots, x_{j-1}, 0, x_{j+1}, \ldots, x_n). \end{aligned} \tag{1.14}$$

f' is identical to f when its variables take on boolean values, yet the degree of x_j is reduced to 1 in f'. Interleaving this operation periodically for every unquantified variable prevents the degree blow-up encountered above, and allows a construction of proof system like Protocol 1.9 for QBF. (The protocol has a "round" for each quantifier and each application of the degree-reduction operation, and the consistency checks $p_i(0)+p_i(1) = p_{i-1}(\alpha_{i-1})$ are replaced with ones to check consistency with Equations (1.12), (1.13), and (1.14).) □

1.4. Additional Topics

1.4.1. Message Complexity

A striking contrast between the interactive proofs for GRAPH NONISOMORPHISM (Protocol 1.5) and **co-NP/PSPACE** (Protocol 1.9) is that the latter requires much more interaction, as measured in the following way:

Definition 1.15 (message complexity[3]). *An interactive protocol (A, B) has* message complexity $m(n)$ *if on every input x and every choice of the random coins for A and B, the number of messages computed before the first* `accept/reject/halt` *message is at most $m(|x|)$.*

The class of languages possessing interactive proofs with constant *message complexity is denoted* **AM**.[4]

It is natural to ask whether more interaction increases the expressive power of interactive proofs. That is, are there languages which have interactive proofs of message complexity $m(n)$ but not $m'(n)$ for some functions m', m? The following result shows that increasing the number of messages by a constant factor does not yield more power:

Theorem 1.16 ([**BM88**]). *For any constant $c \in \mathbb{N}$ and any function $m(\cdot) \geq 2$, the following holds: If L has an interactive proof with message complexity $cm(\cdot)$, then L has an interactive proof with message complexity $m(\cdot)$.*

On the other hand, it is known that interactive proofs with constant message complexity can only prove languages that are low in the polynomial-time hierarchy (specifically, $\mathbf{AM} \subset \Pi_2$) [**BM88**], we have seen that all of **PSPACE** is provable with no restriction on the number of messages (Thm. 1.11). Hence, polynomially many rounds of interaction cannot be reduced to a constant unless $\mathbf{PSPACE} = \Pi_2$. In fact, it is unlikely that such an improvement is possible even for **co-NP**:

Theorem 1.17 ([**BHZ87**]). *If* **co-NP** $\subset$ **AM**, *then the polynomial-time hierarchy collapses (specifically,* $\mathbf{PH} = \Pi_2$*).*

Recall that it is widely believed that the polynomial-time hierarchy does not collapse (cf., the lectures of Steven Rudich in this volume). Since GRAPH NONISOMORPHISM is in **AM** (Protocol 1.5 consists of two rounds), we obtain the following interesting consequence:

Corollary 1.18. GRAPH ISOMORPHISM *is not* **NP**-*complete unless the polynomial-time hierarchy collapses.*

---

[4]The notation **AM** comes from *Arthur–Merlin games*, which was the name given to the type of interactive proofs introduced in [**BM88**]. Arthur–Merlin games are the same as *public-coin interactive proofs*, which we discuss in Section 1.4.2.

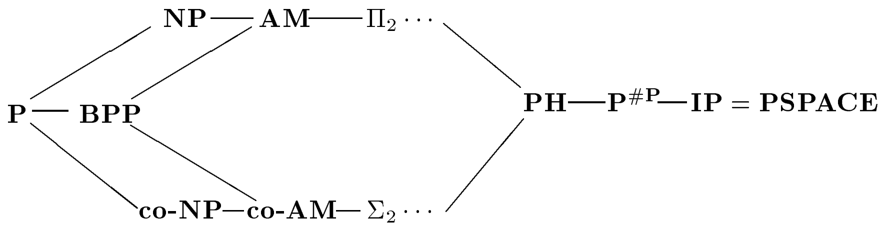

Figure 1. Relation of **IP** and **AM** to other complexity classes. Lines indicate left-to-right inclusion.

Proof. If Graph Isomorphism were **NP**-complete, the Graph Nonisomorphism would be **co-NP**-complete and we would have $\mathbf{co\text{-}NP} \subset \mathbf{AM}$. □

The above proof refers to **NP**-completeness via standard Karp reductions (also known as "many-one" or "mapping" reductions), but it can be easily extended to more general forms of reducibility such as Cook reductions [**Sch88**] (see also [**GG00**]).

1.4.2. Private Coins vs. Public Coins

Recall that it was essential in the proof system for Graph Nonisomorphism (Protocol 1.5) that the verifier's coin tosses are "private," meaning that they are not visible to the prover. In striking contrast, the verifier needs no hidden randomness in the proof systems for **co-NP** (Protocol 1.9) and **PSPACE**. That is, those proof systems satisfy the following definition:

Definition 1.19 (public-coin proofs [**BM88**]). *An interactive proof system is* public coin *if each of the verifier's messages consists of random coin tosses, uniform and independent of the previous messages (except for the last* `accept/reject/halt` *message).*

Since **PSPACE** has a public-coin proof system and **IP** = **PSPACE**, it follows that public-coin interactive proofs are as powerful as private-coin ones. However, there is a stronger (and older) equivalence between private coins and public coins that also preserves message complexity:

Theorem 1.20 ([**GS89**]). *If a language has an interactive proof with message complexity $m(n)$, then it has a* public-coin *interactive proof with message complexity $m(n)$.*

This theorem is very useful in proving results about interactive proofs, since the structured behavior of the verifier in public-coin proofs makes them much easier to analyze and manipulate. Indeed, the proofs of Theorems 1.16 and 1.17 begin by using Theorem 1.20 to reduce to the public-coin case.

Applying Theorem 1.20 to the proof system for Graph Nonisomorphism (Protocol 1.5), we obtain the following consequence:

Corollary 1.21. Graph Nonisomorphism *has a 2-message public-coin interactive proof system.*

One of the exercises involves constructing a 2-message public-coin interactive proof for a problem related to Graph Nonisomorphism (using the same tools that underlie the proof of Theorem 1.20).

1.4.3. The Power of the Prover

Even though the definition of interactive proofs places no computational restrictions on the prover strategy, it is interesting to investigate what power the prover actually needs. If (P, V) is an interactive for a language L, then the complexity of the prover strategy P must, in some sense, be at least the complexity of the language L itself, because one can decide membership in L by simulating the interaction between P and V. The following definition identifies those proof systems for which this lower bound on the prover's complexity is tight.

Definition 1.22 ([**BG94**]). *An interactive proof system (P, V) for a language L is* competitive *if the prover strategy P can be computed in probabilistic polynomial time given a membership oracle for L.*

Which problems have competitive interactive proofs? SATISFIABILITY (and hence every **NP**-complete problem) has a competitive interactive proof, by the well-known fact that using an oracle for deciding SAT, one can actually find satisfying assignments in polynomial time. The GRAPH NONISOMORPHISM proof system (Protocol 1.5) is also competitive, as the prover strategy amounts to deciding GRAPH ISOMORPHISM. With a little more work, it can be verified that the prover in Protocol 1.9 can be implemented using a #**P**-oracle, and hence #**P**-complete problems have competitive interactive proofs. Finally, it follows from one of the exercises that **PSPACE**-complete problems also have competitive interactive proofs. However, it is unlikely that all problems in **IP** have competitive interactive proofs:

Theorem 1.23 ([**BG94**]). *If nondeterministic double-exponential time is not contained in probabilistic double-exponential time, then there is a problem in* **NP** *which has no competitive interactive proof.*

There are a couple of intriguing open problems involving competitive interactive proofs.

Open Problem 1.24. *Do* **co-NP***-complete problems have competitive interactive proofs?*

The best upper bound known on the complexity of a prover for **co-NP** is #**P**, as in Protocol 1.9.

Open Problem 1.25. *Does* GRAPH NONISOMORPHISM *have a* public-coin *competitive interactive proof? More generally, are there any problems for which public-coin interactive proofs require provers with greater complexity than private-coin interactive proofs?*

Recall that there is a transformation which converts private-coin interactive proofs to public-coin ones (Theorem 1.20), but that transformation does not preserve the prover's complexity (and no "black box" transformation can [**Vad00**]).

1.5. Exercises

Exercise 1 (The verifier's randomness is essential)**.** Show that the class of languages possessing interactive proofs with a deterministic verifier is simply **NP**.

Exercise 2 (The prover's randomness is inessential)**.** Show that every language having an interactive proof has one with a deterministic prover.

Exercise 3 (Upper-bounding the power of interaction)**.** Convince yourself that **IP** $\subset$ **PSPACE**. (Hint: What is the complexity of computing the deterministic prover strategy you constructed in Problem 2?)

Exercise 4 (Soundness of GRAPH NONISOMORPHISM interactive pf)**.** Show that if $G_0 = ([n], E_0)$ and $G_1 = ([n], E_1)$ are isomorphic graphs, then $\pi(G_0)$ and $\pi(G_1)$ are identically distributed when π is a uniformly chosen permutation of the vertex set $[n]$.

Exercise 5 (Public-coin lower bound protocol*)**.** A family $\mathcal{H}$ of functions mapping X to Y is called *pairwise independent* if when we choose h uniformly at random from $\mathcal{H}$, the following two conditions hold:

- For all $x \in X$, $h(x)$ is distributed uniformly in in Y.
- For all $x_1 \neq x_2 \in X$, $h(x_1)$ and $h(x_2)$ are independent.

(Efficiently computable pairwise independent families mapping $\{0,1\}^n$ to $\{0,1\}^m$ exist, *e.g.,* the set of functions of the form $h_{A,b}(x) = Ax + b$ where A is an $m \times n$ 0–1 matrix, $b \in \{0,1\}^m$, and all arithmetic is modulo 2.)

1. Let $\mathcal{H}$ be a pairwise independent family of functions mapping X to Y, let $S \subset X$, and let y be any fixed element of Y. Show that
   (a) If $|S| \leq \delta \cdot |Y|$, then $\Pr_{h \leftarrow \mathcal{H}}[\exists x \in S \text{ s.t. } h(x) = y] \leq \delta$
   (b) If $|S| \geq (1/\delta) \cdot |Y|$, then $\Pr_{h \leftarrow \mathcal{H}}[\exists x \in S \text{ s.t. } h(x) = y] \geq 1 - \delta$. (Hint: Use Chebychev's Inequality.)
2. An *automorphism* of a graph is an isomorphism with itself. A graph is *rigid* if it has no automorphisms other than the identity. Use Part (1) to construct a public-coin interactive proof for the language of rigid graphs. (Hint: Let S be the set of 100-tuples of graphs that are isomorphic to the input graph.)

Solution Sketches

Solution 1. A transcript of an interaction in which the verifier accepts constitutes an **NP** proof. Note that the validity of such a transcript (*i.e.,* consistency with the verifier's algorithm) can be checked in poly time.

Solution 2. An "optimal" prover computes each message to maximize the acceptance probability of the verifier given the transcript of the interaction so far. This strategy is deterministic.

Solution 3. We need to show that the maximum possible acceptance probability $p(t)$ of the verifier given the transcript t of the interaction so far can be computed in **PSPACE**. This can be done recursively: If the next move is the prover's, then $p(t) = \max_m p(t \circ m)$ (where we take the maximum over prover messages m). If the next move is the verifier's, then $p(t) = \sum_m q_{t,m} \cdot p(t \circ m)$, where $q_{t,m}$ is the probability that the verifier's next message is m given that the transcript so far is t. Note that $q_{t,m}$ can be computed by enumerating over all the verifier's coin tosses (and discarding those that are not consistent with t.).

Solution 4. Let τ be such that $\tau(G_0) = G_1$. Then for every graph H, the map $\pi \mapsto \tau \circ \pi$ is a bijection between the set of permutations taking G_1 to H and those taking G_0 to H. ($\pi(G_1) = H \Leftrightarrow \pi(\tau(G_0)) = H$.)

Solution 5.

1. (a) This is just a union bound — each $x \in S$ has probability $1/|Y|$ of mapping to y, so the probability that any of them maps to y is at most $|S| \cdot (1/|Y|) \leq \delta$.

   (b) This is an application of Chebychev. Define indicators I_x for the condition $h(x) = y$. We are interested in the probability (over the choice of h) of the event that the sum $M = \sum_{x \in S} I_x$ is greater than 0. Each I_x has expectation $1/|Y|$, so $\mathrm{E}[M] = |S| \cdot (1/|Y|)$. Each I_x has variance $(1 - 1/|Y|) \cdot (1/|Y|) < 1/|Y|$. Since they are pairwise independent, $\mathrm{Var}[M] \leq |S| \cdot (1/|Y|)$. Hence, by Chebychev's Inequality,

$$\Pr[M = 0] \leq \Pr[|M - \mathrm{E}[M]| \geq \mathrm{E}[M]] \leq \frac{\mathrm{Var}[M]}{\mathrm{E}[M]^2} \leq \frac{|Y|}{|S|} \leq \delta.$$

2. The number of graphs isomorphic to G equals $n!$ divided by the number of automorphisms of G, including the identity. (The number of permutations taking G to any H isomorphic to G is exactly the number of automorphisms of G.) Hence, if G has no automorphisms other than the identity then there are $n!$ graphs isomorphic to G, and if G has at least 1 automorphism other than the identity then there are at most $n!/2$ isomorphic to G. Taking 100-tuples amplifies the gap to 2^{100}, and we get the following proof system: The Verifier randomly chooses a hash function h mapping to $\{0,1\}^\ell$ for $\ell \approx \log_2(n!/2^{50})$. The Prover is then supposed to return a 100-tuple of graphs $(G_1, G_2, \ldots, G_{100})$ isomorphic to G such that $h(G_1, \ldots, G_{100}) = 0^\ell$. To prove that these 100 graphs are isomorphic to G, the prover also sends the corresponding isomorphisms. Completeness and soundness follow from the argument above and Part (1).

LECTURE 2
Zero-Knowledge Proofs

Given the importance of proofs in mathematics and computer science, it is natural to ask "What does one learn from a proof?" By definition, upon verifying a proof, one should be convinced that the assertion being proven is true. But a proof can actually reveal much more than that. Indeed, proofs in mathematics are often valued for providing insight in addition to validating a particular theorem. And, at a minimum, it seems inherent in classical proofs that after verifying a proof, one leaves not just with confidence that the assertion is true, but also with the ability to present the same proof to others and convince them of the assertion.

Interactive proofs, however, are not bound by the same limitations as classical proofs. We will see below that it is possible for an interactive proof to be *zero knowledge*, with the verifier learning *nothing* other than than the validity of the assertion being proven. In particular, after verifying such a proof, one does gain the ability to convince someone else of the same statement!

2.1. Definition

It is remarkable that the zero-knowledge property can even be defined in a meaningful and realizable manner. This is accomplished by the *simulation paradigm*: we say that verifier has learned nothing from its interaction with the prover if the verifier can "simulate" its view of the interaction on its own. That is, there should be an efficient probabilistic algorithm, called a *simulator*, whose output distribution is indistinguishable from what the verifier sees when interacting with the prover. Intuitively, this means that the verifier learns nothing since it can run the simulator instead of interacting with the prover.

Definition 2.1 (view of an interactive protocol). *Let (A, B) be an interactive protocol. B's* view *of (A, B) on common input x is the random variable $\langle A, B\rangle(x) = (m_1, \ldots, m_t; r)$ consisting of all the messages $m_1, \ldots, m_t$ exchanged between A and B together with the string r of random bits that B has read during the interaction.*[1]

Definition 2.2 (zero-knowledge proofs [**GMR89**]).
An interactive proof system (P, V) for a language L is said to be zero knowledge *if*

[1] It may seem unnatural that our notation is asymmetric in that it does not allow for indicating A's view of the protocol. However, in these lectures, we will only be interested in B's view (as B corresponds to the verifier in an interactive proof), and thus we have opted for a simpler notation at the expense of generality.

for every probabilistic polynomial-time V^*, there exists a probabilistic polynomial-time simulator S *such that*

$$\{S(x)\}_{x \in L} \qquad \text{and} \qquad \{\langle P, V^* \rangle(x)\}_{x \in L}$$

are computationally indistinguishable.[2] *That is, for every (nonuniform) polynomial-time algorithm D, there is a negligible*[3] *function α such that for all $x \in L$,*

$$|\Pr[D(x, S(x)) = 1] - \Pr[D(x, \langle P, V^* \rangle(x)) = 1]| \le \alpha(|x|).$$

Note that the simulation is only required to be accurate on inputs $x \in L$; that is, when the assertion being proven is true. We wanted the definition to capture the fact that the verifier should learn nothing from the "proof" (which is now actually the strategy for P). For inputs $x \notin L$, there is no "correct" proof (as guaranteed by soundness), so it would be somewhat strange to require that the verifier learns nothing in this case. From a cryptographic point of view, this asymmetry corresponds to the idea that we only wish to protect parties that are behaving honestly; a prover that is trying to prove a false assertion is certainly not.

Another important point about the above definition is that we require the zero-knowledge property to hold even if the verifier follows a strategy V^* that deviates from the specified protocol (provided it is still polynomial time). Clearly, this feature is crucial in cryptographic applications. (Though "honest-verifier zero knowledge," in which a simulator is only required for the specified verifier strategy, is already nontrivial and of complexity-theoretic interest.)

2.2. Zero-knowledge Proofs for NP

Definition 2.2 beautifully captures the intuitive notion of "learning nothing," but of course, the question remains whether nontrivial zero-knowledge proofs exist. Remarkably, every problem having a classical proof also has a zero-knowledge proof.

Theorem 2.3 ([**GMW91**])**.** *Every language in* **NP** *has a zero-knowledge proof (assuming one-way functions*[4] *exist).*

With this theorem, zero-knowledge proofs gain vast applicability in cryptography, where it often arises that one party wishes to convince others of some "**NP** assertion" without leaking unnecessary information. For example, zero-knowledge proofs can be used to make protocols robust against cheating parties: participants in the protocol can prove to each other that their actions are consistent with the specified protocol without comprising any of their "secret" information (*e.g.,* their encryption keys) [**Yao86, GMW87**]. They can also be used to construct "identification schemes," whereby one party can "prove" her identity to others without leaking any information that can later be used to impersonate her [**FFS88**].

To prove Theorem 2.3, it suffices to give a zero-knowledge proof for a single **NP**-complete problem. We will use GRAPH 3-COLORING. A *3-coloring* of a graph $G = ([n], E)$ is an assignment $C : [n] \to \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$ (for "Red," "Green," and

[2] See Oded Goldreich's lecture notes in this volume for a detailed discussion of computational indistinguishability. The definition we need differs from the one there in two main respects: the ensembles are indexed by strings in a language rather than all natural numbers, and we allow the distinguisher to be nonuniform (*i.e.,* a circuit).

[3] A function $\alpha : \mathbb{N} \to [0, 1]$ is *negligible* if for every (positive) polynomial p, $\alpha(n) \le 1/p(n)$ for all sufficiently large n.

[4] See the lecture notes of Goldreich in this volume for the definition of one-way functions

"Blue") such that no pair of adjacent vertices are assigned the same color. GRAPH 3-COLORING is the language

$$3\mathrm{COL} = \{G : G \text{ is 3-colorable}\},$$

and it is known to be **NP**-complete (cf., [**Pap94**]).

2.2.1. A "Physical" Protocol

The zero-knowledge proof for GRAPH 3-COLORING is based on the observation that the classical proof can be broken into "pieces" and randomized in such a way that (a) the entire proof is valid if and only if every piece is valid, yet (b) each piece reveals nothing on its own. For GRAPH 3-COLORING, the classical proof is a three-coloring of the graph, and the pieces are the restriction of the coloring to the individual edges: (a) An assigment of colors to vertices of the graph is a proper 3-coloring if and only if the endpoints of every edge have distinct colors, yet (b) if the three colors are randomly permuted, then the colors assigned to the endpoints of any particular edge are merely a random pair of distinct colors and hence reveal nothing.

In Protocol 2.4, we show how to use the above observations to obtain a zero-knowledge proof for GRAPH 3-COLORING which makes use of "physical" implements — namely opaque, lockable boxes. We will later obtain the final proof system by using an appropriate "digital" (*i.e.,* mathematical) primitive which emulates the properties of opaque boxes used.

Protocol 2.4: "Physical" Proof System (P, V) for GRAPH 3-COLORING

Input: A graph $G = ([n], E)$

1. P: Let C be any canonical 3-coloring of G (*e.g.,* the lexicographically first one). Let π be a uniformly selected permutation of $\{R, G, B\}$. Let $C' = \pi \circ C$.
2. P: For every vertex $v \in [n]$, place $C'(v)$ inside a box B_v, lock the box using a key K_v, and send the box B_v to V.
3. V: Uniformly select an edge $(x, y) \in E$ and send (x, y) to P.
4. P: Send the keys K_x and K_y to V.
5. V: Unlock the boxes B_x and B_y, and accept if the colors inside are different.

We now explain why this protocol works. The following "proof" should only be taken as motivation for the final protocol, and the reader should not be disturbed by ambiguities resulting from the fact that we haven't precisely defined this "physical" model.

"Proposition" 2.5. *Protocol 2.4 is a "zero-knowledge proof" for* 3COL.

"Proof". For completeness, first observe that if C is a proper 3-coloring of G then so is C'. Thus, no matter which edge $(x, y) \in E$ the verifier selects, the colors $C'(x)$ and $C'(y)$ inside boxes B_x and B_y will be different. Therefore, the verifier accepts with probability 1 when $G \in 3\mathrm{COL}$.

For soundness, consider the colors inside the boxes sent by the prover in Step 2 as assigning a color to each vertex of G. If G is not 3-colorable, then it must be the case that for some $(x, y) \in E$, B_x and B_y contain the same color. So the verifier will reject with probability at least $1/|E|$. By repeating the protocol $|E|+1$ times, the probability that the verifier accepts on $G \notin$ 3COL will be reduced to $(1 - 1/|E|)^{|E|+1} < 1/3$.

To argue that Protocol 2.4 is "zero knowledge," let's consider what a verifier "sees" in an execution of the protocol (when the graph is 3-colorable). The verifier sees n boxes $\{B_v\}$, all of which are locked and opaque, except for a pair B_x, B_y corresponding to an edge in G. For that pair, the keys K_x and K_y are given and the colors $C'(x)$ and $C'(y)$ are revealed. Of all this, only $C'(x)$ and $C'(y)$ can potentially leak knowledge to the verifier. However, since the coloring is randomly permuted by π, $C'(x)$ and $C'(y)$ are simply a (uniformly) random pair of distinct colors from $\{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$, and clearly this is something the verifier can generate on its own.

In this intuitive argument, we have reasoned as if the verifier selects the edge (x, y) in advance, or at least independently of the permutation π. This would of course be true if the verifier follows the specified protocol and selects the edge randomly, but the definition of zero knowledge requires that we also consider cheating verifier strategies whose edge selection may depend on the messages previously received from the prover (*i.e.,* the collection of boxes). However, the perfect opaqueness of the boxes guarantees that the verifier has no information about their contents, so we can indeed view (x, y) as being selected in advance by the verifier, prior to receiving any messages from the prover. □

2.2.2. The "Digital" Protocol

In order to obtain a "digital" (*i.e.,* mathematical) zero-knowledge proof for GRAPH 3-COLORING, we will replace the opaque boxes with a cryptographic primitive that retains the essential features of the boxes: We should be able "lock" objects (*i.e.,* strings) into "boxes" (again, strings) in such a way that:

1. The locked box completely hides the object locked within it (to maintain the zero-knowledge property).
2. A "key" to open a box and verify its contents can be given (to implement Step 4).
3. The contents of a locked box cannot be changed (to maintain soundness).

The following definition captures the above three properties.

Definition 2.6 (commitment schemes — simplified)). *A* commitment scheme *is a polynomial-time algorithm* Commit *which takes a* message m *and a (random)* key K *and produces a* commitment $B = \mathrm{Commit}(m; K)$. *For a given m, the distribution of B over a uniformly chosen key $K \in \{0,1\}^k$ is denoted* $\mathrm{Commit}_k(m)$. Commit *must satisfy the following properties:*

1. *(unambiguity) For any $m \neq m'$, the set of commitments to m is disjoint from the set of commitments to m'. That is, there do not exist K, K' such that* $\mathrm{Commit}(m; K) = \mathrm{Commit}(m; K')$.
2. *(secrecy) For any m, m', commitments to m and m' are computationally indistinguishable. That is, for every (nonuniform) polynomial-time algorithm*

D, there is a negligible function α such that

$$|\Pr[D(\text{Commit}_k(m)) = 1] - \Pr[D(\text{Commit}_k(m'))]| \leq \alpha(k).$$

Note that a commitment B can indeed be "opened" by providing the corresponding message m and key K, and this can be verified by checking that $B = \text{Commit}(m; K)$.

Commitment schemes meeting the above definition can be construct from any one-way permutation[5] (exercise). There is a more general definition of commitment schemes which allows interaction (cf., [**Gol00**]), and commitment schemes meeting the more general definition exist if and only if one-way functions exist [**HILL99, Nao91**].

Replacing the boxes in Protocol 2.4 with a commitment scheme yields the "digital" zero-knowledge proof for GRAPH 3-COLORING given in Protocol 2.7.

Protocol 2.7: "Digital" Proof System (P, V) for GRAPH 3-COLORING

Input: A graph $G = ([n], E)$

1. P: Let C be any canonical 3-coloring of G (*e.g.,* the lexicographically first one). Let π be a uniformly selected permutation of $\{R, G, B\}$. Let $C' = \pi \circ C$.
2. P: For every vertex $v \in [n]$, choose K_v uniformly in $\{0,1\}^n$, let $B_v = \text{Commit}(C'(v); K_v)$, and send B_v to V.
3. V: Uniformly select an edge $(x, y) \in E$ and send (x, y) to P.
4. P: Send K_x, K_y, $C'(x)$, and $C'(y)$ to V.
5. V: Accept if $B_x = \text{Commit}(C'(x); K_x)$ and $B_y = \text{Commit}(C'(y); K_y)$, and $C'(x) \neq C'(y)$.

2.2.3. Proof of Correctness

We now prove the correctness of Protocol 2.7, establishing Theorem 2.3.[6]

Proposition 2.8. *Protocol 2.7 is a zero-knowledge proof system for* GRAPH 3-COLORING.

Proof. Completeness and soundness are proved as they were for "Proposition" 2.5, using the unambiguity property of commitment schemes to establish soundness.

The zero-knowledge property follows the same intuition as in the physical protocol — all the verifier sees is a random pair of distinct colors, together with the unopened commitments. A random pair of distinct colors is something the verifier can generate on its own, and the secrecy property of the commitment scheme should imply that the verifier learns nothing from the unopened commitments. Based on this intuition, it is straightforward to simulate the verifier's view when the verifier follows the specified protocol: the simulator can randomly select an edge $(x, y) \in E$,

[5] One-way permutations are the same objects referred to as "one-to-one one-way functions" in Goldreich's lecture notes.

[6] Except for the fact that we assume the existence of a commitment scheme in the simplified sense of Definition 2.6, and this is apparently stronger than assuming the existence of one-way functions.

construct B_x and B_y as commitments to a random pair of distinct colors, and construct the remaining commitments as commitments to arbitrary colors (since they need not be opened).

However, for cheating verifiers, this setting presents an additional subtlety not present in the physical protocol. Unlike boxes, commitments do not always "look the same" — they vary as a function of their contents and the key. A cheating verifier can select the edge (x, y) in a way that depends on the commitment. Thus, unlike the physical setting, the simulator cannot determine in advance which edge (x, y) the verifier will select and then place a random pair of distinct colors in B_x and B_y. Instead, the simulator will randomly "guess" which edge the cheating verifier will select, and later check this by running the verifier algorithm. We will argue that the simulator succeeds with noticeable probability ($\approx 1/|E|$), and hence polynomially many trials will yield success with all but negligible probability. A simulator S^{V^*} (for a cheating verifier V^*) designed according to this intuition is given in Algorithm 2.9.

Algorithm 2.9: Simulator S^{V^*} for Protocol 2.7

Input: A graph $G = ([n], E)$, and a cheating verifier algorithm V^*

1. Uniformly select an edge $(x, y) \in E$.
2. Define a coloring $C' : [n] \to \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$ as follows: Select $(C'(x), C'(y))$ uniformly among the distinct pairs from $\{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$, and for $v \notin \{x, y\}$, set $C'(v) = \mathrm{R}$.
3. For every $v \in V$, choose K_v uniformly in $\{0,1\}^n$ and let $B_v = \mathrm{Commit}(C'(v); K_v)$.
4. Run V^* to determine which edge (x^*, y^*) it would select when sent all the B_v's. That is, uniformly select coin tosses r for V^* and let $(x^*, y^*) = V^*(G, \{B_v\}; r)$.
5. If $(x^*, y^*) \neq (x, y)$, output `fail`. Otherwise, output $(\{B_v\}_{v \in V}, (x, y), (K_x, K_y, C'(x), C'(y)); r)$.

Claim 2.10. *For any probabilistic polynomial-time V^*, there is a negligible function α such that on any input $G = ([n], E)$,*

1. *$S^{V^*}(G)$ succeeds with probability at least $1/|E| - \alpha(n)$.*
2. *The output distribution of $S^{V^*}(G)$, conditioned on success, is computationally indistinguishable from $\langle P, V^* \rangle(G)$.*

In order to prove Claim 2.10, it will be convenient to consider a modification of the distribution $\langle P, V^* \rangle(G)$ that incorporates a failure probability:

Distribution $\langle P, V^* \rangle^{\mathtt{f}}(G)$: Choose (x, y) uniformly from E. Sample $\mathtt{view} = (\{B_v\}_{v \in V}, (x^*, y^*), (K_{x^*}, K_{y^*}, C'(x^*), C'(y^*)); r)$ according to $\langle P, V^* \rangle(G)$. If $(x^*, y^*) \neq (x, y)$, output `fail`. Otherwise, output `view`.

$\langle P, V^* \rangle^{\mathtt{f}}(G)$ succeeds with probability exactly $1/|E|$ (since (x, y) is independent of (x^*, y^*)), and conditioned on success, its output distribution is identical to $\langle P, V^* \rangle(G)$. Thus, Claim 2.10 is reduced to showing that $S^{V^*}(G)$ is computationally indistinguishable from $\langle P, V^* \rangle^{\mathtt{f}}(G)$. We will prove this using the secrecy property of the commitment scheme. More precisely, we will argue that if $S^{V^*}(G)$ could be distinguished from $\langle P, V^* \rangle^{\mathtt{f}}(G)$, then the following two distributions would be distinguishable.

Distribution RRR**:** Output $3n$ independent commitments to R.

Distribution RGB**:** Output n independent commitments to R, followed by n independent commitments to B, followed by n independent commitments to G.

(Above, all commitments are using uniformly selected keys of length n, *i.e.,* $\mathrm{Commit}_n(\cdot)$.) Distributions RRR and RGB are computationally indistinguishable by the secrecy of the commitment scheme and a "hybrid argument" (cf., the lecture notes of Oded Goldreich in this volume).

To perform the desired reduction, we will give a (nonuniform) polynomial-time algorithm T which "transforms" Distributions RRR and RGB into $S^{V^*}(G)$ and $\langle P, V^* \rangle^{\mathtt{f}}(G)$, respectively. Thus T can be used to transform a distinguisher between the latter pair of distributions into a distinguisher between the former pair. T will have the graph $G = ([n], E)$ and the coloring C used by the prover "hardwired in"; this is why we need it to be nonuniform.

Algorithm 2.11: Transforming Algorithm T

Input: A sequence of $3n$ commitments

$$(B_1^{\mathrm{R}}, B_2^{\mathrm{R}}, \ldots, B_n^{\mathrm{R}}, B_1^{\mathrm{G}}, \ldots, B_n^{\mathrm{G}}, B_1^{\mathrm{B}}, \ldots, B_n^{\mathrm{B}})$$

Nonuniformity: A 3-colorable graph $G = ([n], E)$, a 3-coloring C of G, and a cheating verifier algorithm V^*

1. Uniformly select an edge $(x, y) \in E$.
2. Let π be a uniformly selected permutation of $\{R, G, B\}$. Let $C' = \pi \circ C$.
3. Choose K_x and K_y uniformly in $\{0,1\}^n$. Let $B_x = \mathrm{Commit}(C'(x); K_x)$, $B_y = \mathrm{Commit}(C'(y); K_y)$.
4. For $v \notin \{x, y\}$, Let $B_v = B_v^{C'(v)}$.
5. Uniformly select coin tosses r for V^* and let $(x^*, y^*) = V^*(G, \{B_v\}; r)$.
6. If $(x^*, y^*) \neq (x, y)$, output `fail`. Otherwise, output $(\{B_v\}_{v \in V}, (x, y), (K_x, K_y, C'(x), C'(y)); r)$.

The transforming algorithm T is given in Algorithm 2.11. It can be verified by inspection that when T is fed Distribution RGB, its output distribution is exactly $\langle P, V^* \rangle^{\mathtt{f}}(G)$. On the other hand, when T is fed Distribution RRR, its output distribution is identical to that of the simulator $S^{V^*}(G)$ (since when C is a proper 3-coloring, $C'(x)$ and $C'(y)$ are indeed a random pair of distinct colors). This proves that $S^{V^*(G)}$ is computationally indistinguishable from $\langle P, V^* \rangle^{\mathtt{f}}(G)$, which in turn establishes Claim 2.10 and Proposition 2.8. $\square$

2.2.4. Remarks

A few remarks about the proofs of Theorem 2.3 and Proposition 2.8 are in order. First, although the definition of interactive proofs allows a computationally unbounded prover, the strategy of the prover in Protocol 2.7 can actually be implemented in *polynomial time* when given an **NP** witness (*i.e.,* a 3-coloring of the graph). This property is crucial in cryptographic applications of zero-knowledge proofs, where we typically want the computations required of all parties to be efficient (though we may wish for security against computationally unbounded adversaries).

The simulation is another place in which the proof gives something stronger than required by the definition. The definition only requires that for every verifier strategy V^*, *there exists* a simulator. However, Algorithm 2.9 gives a single "universal" simulator S which works for all verifier strategies V^*, using this verifier strategy only as a "black box." That is, the simulator only requires access to the input-output behavior of V^*, and not the program which computes it. All known zero-knowledge proofs are demonstrated correct using such *universal black-box simulation*, and it is difficult to imagine how one would prove the zero-knowledge property in any other way. On the other hand, there are several limitations on the efficiency of black-box zero-knowledge proofs that are not known to hold for the general definition, so there is some motivation to seek alternatives to this paradigm.

We also remark on the use of **NP**-completeness in the proof of Theorem 2.3. **NP**-completeness results are most often thought of as "negative" statements, as they give evidence of a problem's intractability. Here, however, we have used **NP**-completeness in a "positive" way — to reduce the task of proving something about all of **NP** to the task of proving something about a single **NP**-complete problem, namely GRAPH 3-COLORING. (There was a similar positive use of completeness in the proofs of Theorems 1.6 and 1.11.)

Finally, we mention a result showing that the seemingly strong zero-knowledge condition actually does not limit the expressive power of interactive proofs *at all*:

Theorem 2.12 ([**IY87, BGG$^+$88**]). *Every problem in* **IP** *has a zero-knowledge proof (assuming one-way functions exist).*

While it is a substantial strengthening of Theorem 2.3 from a complexity-theoretic viewpoint, Theorem 2.12 does not yield much more utility for cryptographic protocols. The reason is that the crucial property guaranteed by the proof of Theorem 2.3 — that the prover can be implemented in polynomial time given an **NP** witness — cannot be extended to Theorem 2.12 for this property does not even make sense for problems outside **NP**.

2.3. Additional Topics

2.3.1. Composition of Zero-Knowledge Proofs

When presenting the GRAPH 3-COLORING proof system above, we cavalierly said "repeat the protocol several times to reduce the error probability." While it is true that repetitions do work for reducing the error probability, their effect on the zero-knowledge property is more subtle. To explain the issue in more detail, we need to be more precise about what we mean by "repetitions." Two natural interpretations are:

Sequential Composition: The k executions of the proof system are performed one after another. So if the original proof system has message complexity m, the new proof system has message complexity km.

Parallel Composition: The k executions of the proof system are carried out all at once, "in lock step." That is, the message complexity of the proof system remains the same, and each message of the new proof system consists of a k-tuple of messages in the original proof system.

Of these two, the zero-knowledge property is only preserved under sequential composition, and even that requires a modification of Definition 2.2 to allow the verifier an "auxiliary input" (to model the verifier's state after prior interactions) (cf., [**FS90, GO94, GK96b**]). The fact that zero knowledge is not closed under parallel composition makes it difficult to construct zero-knowledge proofs which simultaneously have low message complexity and negligible error probability. Furthermore, there are inherent limitations on constructing such zero-knowledge proofs, at least using black-box simulation:

Theorem 2.13 ([**GK96b**]). *Only problems in* **BPP** *have 3-message black-box simulation zero-knowledge proofs with negligible error probabilities (in the completeness and soundness conditions). For public-coin proofs, the same result holds for any constant message complexity.*

Still, using private coins and a stronger complexity assumption, it is known how to construct constant-message zero-knowledge proofs.

Theorem 2.14 ([**GK96a**]). *If a family of "claw-free permutations" exists, then* **NP** *has 5-message zero-knowledge proofs.*

Recently, much attention has focused on the behavior of the zero-knowledge property under more general, "adversarial" forms of repetition to model situations that can arise in cryptographic applications. One object of study along these lines has been *concurrent zero knowledge* [**DNS98**], which asks for protocols whose zero-knowledge property is preserved even when many of them are executed at the same time and the verifier can adversarially determine how the steps of the various protocols are interleaved. Such a situation could arise, for example, when zero-knowledge proofs are being employed in a distributed environment such as the Internet. An even stronger requirement that has been studied is *resettable zero knowledge* [**CGGM00**], which asks that the zero-knowledge property be preserved even if the verifier can force the prover to execute the protocol many times using the same coin tosses. This might be a realistic attack on physical implementations of zero-knowledge proofs, where the prover is implemented on, say, a smart card.

Given that standard zero-knowledge proofs are not closed under even parallel composition, it is not surprising that the construction of message-efficient concurrent and resettable zero-knowledge proofs is quite difficult. To overcome these difficulties, some researchers have considered augmenting the model of interaction with additional features such as "timing" or "public keys" [**DNS98, Dam99, CGGM00**]. Other researchers have investigated these notions in the standard interactive model, attempting to determine the minimal message complexity needed for **NP** to have concurrent or resettable zero-knowledge proofs. While there has been considerable progress, at the time of these lectures there is still a significant gap between the known upper bounds [**RK99, CGGM00, KP00**] and lower bounds [**KPR98, Ros00**] (which are stronger than those given by Theorem 2.13).

2.3.2. Perfect and Statistical Zero Knowledge

The definition of zero-knowledge proofs (Definition 2.2) requires the simulator's output to be computationally indistinguishable from the verifier's view of the interaction. Here, we will consider two "information-theoretic" strengthenings of this requirement:

Perfect zero knowledge: The simulator's output distribution is *identical* to the verifier's view of the interaction.

Statistical zero knowledge: The simulator's output distribution is *statistically close* to the verifier's view. More precisely, their statistical difference[7] is bounded by a negligible function of the input length.

The class of languages possessing statistical (resp., perfect) zero-knowledge proofs is denoted **SZK** (resp., **PZK**). For contrast, zero-knowledge proofs in the sense of Definition 2.2 are often referred to as *computational zero knowledge* and the class of languages possessing them is denoted **CZK**. Clearly, $\mathbf{PZK} \subset \mathbf{SZK} \subset \mathbf{CZK}$.

Statistical and perfect zero-knowledge proofs provide much stronger "security" guarantees than computational ones, in that the zero-knowledge condition is meaningful even for verifiers with unbounded computational power. Surprisingly, these stronger requirements can be met, and perfect zero-knowledge proofs are known to exist for a number of nontrivial problems of complexity-theoretic and cryptographic interest: Quadratic Residuosity and Nonresiduosity [**GMR89**], Graph Isomorphism and Nonisomorphism [**GMW91**], the Discrete Logarithm problem [**GK93**], and approximate versions of the Shortest Vector and Closest Vector problems in lattices [**GG00**].

Despite containing these problems believed to be hard, there are are also strong upper bounds on the complexity of **SZK**:

Theorem 2.15 ([**For89, AH91**]). $\mathbf{SZK} \subset \mathbf{AM} \cap \mathbf{co}\text{-}\mathbf{AM}$.

By Theorem 1.17, this means that it is unlikely that **SZK** contains **NP**-hard problems. This puts **SZK** in an intriguing region in complexity theory — lying somewhere between the tractable problems (*i.e.,* **BPP**) and the **NP**-hard ones. This is striking contrast to **CZK** which equals **PSPACE** if one-way functions exist (by Theorems 1.11 and 2.12).

Recently, there has been substantial progress in improving our understanding of statistical zero knowledge. Here, we mention two results which have shed more light on the the complexity of the class **SZK**.

Theorem 2.16 ([**Oka00**]). **SZK** *is closed under complement.*

This result is surprising because of the asymmetric definition of **SZK**. There is no *a priori* reason to believe that if one can prove that a statement is true in zero knowledge then one should also be able to prove that it is false in zero knowledge; this is similar to the intuition that underlies our belief that $\mathbf{NP} \neq \mathbf{co}\text{-}\mathbf{NP}$. In fact, **IP** and **CZK** are also closed under complement (assuming one-way functions exist for **CZK**), but those are a trivial consequences of the more dramatic result showing that they are equal to **PSPACE** (Thms. 1.11 and 2.12).

[7]The statistical difference between two probability distributions X and Y on a set D is $\max_{S \subset D} |\Pr[X \in S] - \Pr[Y \in S]|$.

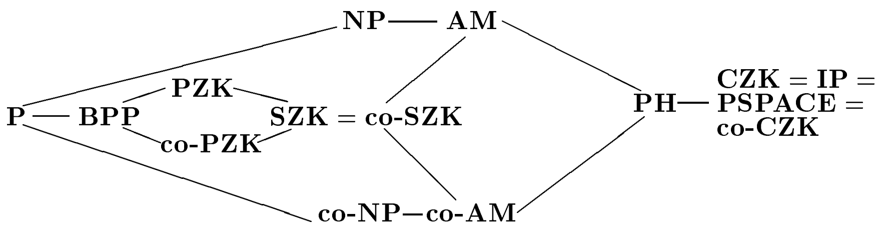

Figure 1. Relation of **PZK**, **SZK**, and **CZK** to other complexity classes (assuming one-way functions exist). Lines indicate left-to-right inclusion.

Theorem 2.17 ([**SV97, GV99**]). **SZK** *has two complete problems, called* Statistical Difference *and* Entropy Difference. *(These problems essentially amount to approximating the statistical difference or the difference in entropies between two distributions specified by algorithms (circuits) which sample from them.)*

These problems give a characterization of **SZK** that makes no reference to interaction or zero knowledge, and provide further evidence that **SZK** captures a rich and natural class of computational problems. Furthermore, they have proven to be very useful for obtaining general results about **SZK**, as they reduce questions about the entire class to ones about a single problem. Thus, we see more “positive” uses of completeness in this area.

There are many open problems regarding statistical zero knowledge (cf., [**Vad99**]), but here we just mention two.

Open Problem 2.18. *Does* **SZK** = **PZK**?

Open Problem 2.19. *Find a complete problem for* **SZK** *that is combinatorial or number-theoretic (rather than statistical) in nature.*

2.4. Exercises

Exercise 1 (Commitment schemes)**.** Construct a commitment scheme from any one-way permutation (which cannot be inverted by polynomial-sized circuits).[8]

Exercise 2 (Honest-verifier zero knowledge)**.** An *honest-verifier* zero-knowledge proof is one in which the simulation condition is only required to hold for the specified verifier V (rather than all polynomial-time verifiers V^*).

1. Show that the interactive proof for Graph Nonisomorphism given in lecture is honest-verifier (perfect) zero knowledge.
2. Construct a similar honest-verifier perfect zero-knowledge proof system for Quadratic Nonresiduosity, *i.e.,* the language

$$\mathrm{QNR} = \{(n, x) : \text{there is no } y \text{ such that } y^2 = x \pmod{n}\}.$$

Exercise 3 (Perfect zero knowledge)**.** Exhibit a perfect zero-knowledge proof for Quadratic Residuosity, *i.e.,* the complement of Quadratic Nonresiduosity from Problem 2. (You should exhibit a simulator even for cheating verifiers. The

[8] The key-length in your construction may depend on the message length, although technically Definition 2.6 does not allow such a dependence. (This dependency can be removed using a pseudorandom generator, as defined in the lecture notes of Goldreich in this volume.)

simulation may fail with probability, say, 1/2, as long as its output distribution is correct conditioned on non-failure.)

Exercise 4 (Resettable zero knowledge)**.** Informally, a zero-knowledge proof is *resettable* if it remains zero knowledge even when the verifier can force the prover to use the same coin tosses in polynomially many interactions. Find a zero-knowledge proof which is not resettable (under a reasonable complexity assumption).

Solution Sketches

Solution 1. Let B be a hard-core predicate for a one-way permutation f. To commit to a bit b, choose x at random and output $(f(x), B(x) \oplus b)$. Unambiguity follows because f is one-to-one. And secrecy follows from the fact that $(f(x), B(x))$ is indistinguishable from uniform and hence also from $(f(x), B(x) \oplus 1)$. (See the construction of pseudorandom generators which stretch by 1 bit in Goldreich's lecture notes.) To commit to a long message m, apply this commitment scheme to each bit of m (using independently chosen x's for each bit). The indistinguishability of Commit(m) and Commit(m') for all m, m' follows from a hybrid argument reducing to secrecy of the 1-bit commitment scheme. (The reduction will need to have the messages m, m' hardwired in; this is why we need to work with circuits rather than uniform adversaries.)

Solution 2. For Graph Nonisomorphism, the simulator just mimics the verifier and produces a transcript in which the prover answers correctly (which happens w.p. 1 in the real interaction on YES instances). The proof system for Quadratic Nonresiduosity is as follows: the verifier chooses a random $r \in \mathbb{Z}_n^*$ and flips a coin $b \in \{0,1\}$. If $b = 0$, she sends the prover r^2 and if $b = 1$, she sends the prover $x \cdot r^2$. The prover must guess b. When x is a quadratic nonresidue, the distributions r^2 and xr^2 are disjoint; otherwise, they are identical. The analysis proceeds as for Graph Nonisomorphism.

Solution 3. On input (n, x), the prover sends the verifier a random square s modulo n, and then the verifier asks the prover to return a square root of either s or sx; the prover chooses one of the possible square roots at random. If x is a square, this will always be possible. If x is a nonsquare, at most 1 of x, sx has a square root, so the verifier will reject with probability at least 1/2. The simulator chooses r uniformly in $\mathbb{Z}_n^*$, randomly guesses the verifier's challenge, and accordingly sends either $s = r^2$ or $s = r^2/x$ as the prover's message. It then runs the verifier V^* to find out whether it guessed the challenge correctly. If yes, it uses r as the prover's last message. If not (which happens w.p. 1/2), it fails. It can be verified that conditioned on non-failure, the output distribution is identical to the real interaction.

Solution 4. The proof system for Graph 3-Coloring given in lecture is an example. By making the prover run n with the same coin tosses and querying an edge touching a new vertex each time, the verifier can learn a 3-coloring of the graph. Hence this cannot be simulated in poly-time unless $\mathbf{NP} \subset \mathbf{BPP}$.

SUGGESTIONS FOR FURTHER READING

These lectures were *not* intended to be comprehensive surveys of the areas covered. The "additional topics" sections in particular were designed to give a small sample of recent research directions and open problems, and are largely a reflection of the author's own interests. Here we mention some places where the interested reader can learn more about this area.

[**Gol99**, Ch. 2] contains a broad survey of probabilistic proof systems, including variants of interactive and zero-knowledge proofs not treated in these lectures. More details of proof that **IP** = **PSPACE** (Thm. 1.11) can be found in [**Sip97**, 10.4]. An entertaining account of the ideas leading up to that theorem can be found in [**Bab90**]. Zero-knowledge proofs are covered in great depth and detail in [**Gol00**, Ch. 4]. A unified treatment of the recent work on statistical zero knowledge can be found in [**Vad99**].

BIBLIOGRAPHY

[AH91] William Aiello and Johan Håstad. Statistical zero-knowledge languages can be recognized in two rounds. *Journal of Computer and System Sciences*, 42(3):327–345, June 1991.

[AK97] Vikraman Arvind and Johannes Köbler. On resource-bounded measure and pseudorandomness. In *Proceedings of the 17th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 235–249. LNCS 1346, Springer-Verlag, 1997.

[Bab90] László Babai. E-mail and the unexpected power of interaction. In *Proceedings, Fifth Annual Structure in Complexity Theory Conference*, pages 30–44, Barcelona, Spain, 8–11 July 1990. IEEE Computer Society Press.

[BM88] László Babai and Shlomo Moran. Arthur-Merlin games: A randomized proof system and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[BG94] Mihir Bellare and Shafi Goldwasser. The complexity of decision versus search. *SIAM Journal on Computing*, 23(1):97–119, 1994.

[BGG$^+$88] Michael Ben-Or, Oded Goldreich, Shafi Goldwasser, Johan Håstad, Joe Kilian, Silvio Micali, and Phillip Rogaway. Everything provable is provable in zero-knowledge. In S. Goldwasser, editor, *Advances in Cryptology—CRYPTO '88*, volume 403 of *Lecture Notes in Computer Science*, pages 37–56. Springer-Verlag, 1990, 21–25 August 1988.

[BHZ87] Ravi B. Boppana, Johan Håstad, and Stathis Zachos. Does co-NP have short interactive proofs? *Information Processing Letters*, 25:127–132, 1987.

[CGGM00] Ran Canetti, Oded Goldreich, Shafi Goldwasser, and Silvio Micali. Resettable zero-knowledge (extended abstract). In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 235–244, Portland, OR, May 2000. ACM.

[Dam99] Ivan Damgård. Concurrent zero-knowledge is easy in practice. Technical Report 1999/014, Cryptology ePrint Archive, `http://eprint.iacr.org/`, 1999.

[DNS98] Cynthia Dwork, Moni Naor, and Amit Sahai. Concurrent zero-knowledge. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, pages 409–418, Dallas, 23–26 May 1998.

[FFS88] Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, 1988.

[FS90] Uriel Feige and Adi Shamir. Witness indistinguishable and witness hiding protocols. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 416–426, Baltimore, Maryland, 14–16 May 1990.

[For89] Lance Fortnow. The complexity of perfect zero-knowledge. In Silvio Micali, editor, *Advances in Computing Research*, volume 5, pages 327–343. JAC Press, Inc., 1989.

[Gol99] Oded Goldreich. *Modern Cryptography, Probabilistic Proofs, and Pseudorandomness*. Number 17 in Algorithms and Combinatorics. Springer-Verlag, 1999.

[Gol00] Oded Goldreich. *Foundations of Cryptography (Volume 1 — Basic Tools)*, 2000. To be published by Cambridge University Press. Preliminary versions and further information available from `http://www.wisdom.weizmann.ac.il/~oded/foc-book.html`.

[GG00] Oded Goldreich and Shafi Goldwasser. On the limits of nonapproximability of lattice problems. *Journal of Computer and System Sciences*, 60(3):540–563, 2000.

[GK96a] Oded Goldreich and Ariel Kahan. How to construct constant-round zero-knowledge proof systems for NP. *Journal of Cryptology*, 9(3):167–190, 1996.

[GK96b] Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):169–192, February 1996.

[GK93] Oded Goldreich and Eyal Kushilevitz. A perfect zero-knowledge proof system for a problem equivalent to the discrete logarithm. *Journal of Cryptology*, 6:97–116, 1993.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, 25–27 May 1987.

[GMW91] Oded Goldreich, Silvio Micali, and Avi Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. *Journal of the ACM*, 38(1):691–729, 1991.

[GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *Journal of Cryptology*, 7(1):1–32, Winter 1994.

[GV99] Oded Goldreich and Salil Vadhan. Comparing entropies in statistical zero-knowledge with applications to the structure of SZK. In *Proceedings of the Fourteenth Annual IEEE Conference on Computational Complexity*, pages 54–73, Atlanta, GA, May 1999. IEEE Computer Society Press.

[GMR89] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, February 1989.

[GS89] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In Silvio Micali, editor, *Advances in Computing Research*, volume 5, pages 73–90. JAC Press, Inc., 1989.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396 (electronic), 1999.

[IY87] Russell Impagliazzo and Moti Yung. Direct minimum-knowledge computations (extended abstract). In Carl Pomerance, editor, *Advances in Cryptology—CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 40–51. Springer-Verlag, 1988, 16–20 August 1987.

[KP00] Joe Kilian and Erez Petrank. Concurrent zero-knowledge in polylogarithmic rounds. Technical Report 2000/013, Cryptology ePrint Archive, `http://eprint.iacr.org/`, 2000.

[KPR98] Joe Kilian, Erez Petrank, and Charles Rackoff. Lower bounds for zero-knowledge on the internet. In *39th Annual Symposium on Foundations of Computer Science*, Palo Alto, California, 8–11November 1998. IEEE.

[KvM99] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In *Proceedings of the Thirty-first Annual ACM Symposium on Theory of Computing*, pages 659–667, Atlanta, 1–4 May 1999.

[LFKN92] Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, October 1992.

[MV99] Peter Bro Miltersen and N. V. Vinodchandran. Derandomizing Arthur–Merlin games using hitting sets. In *40th Annual Symposium on Foundations of Computer Science*, pages 71–80, New York City, New York, 17–19October 1999. IEEE.

[Nao91] Moni Naor. Bit commitment using pseudorandomness. *Journal of Cryptology*, 4(2):151–158, 1991.

[Oka00] Tatsuaki Okamoto. On relationships between statistical zero-knowledge proofs. *Journal of Computer and System Sciences*, 60(1):47–108, February 2000.

[Pap94] Christos H. Papadimitriou. *Computational Complexity.* Addison–Wesley, 1994.

[RK99] Ransom Richardson and Joe Kilian. On the concurrent composition of zero-knowledge proofs. In *Advances in cryptology—EUROCRYPT '99 (Prague)*, pages 415–431. Springer, Berlin, 1999.

[Ros00] Alon Rosen. A note on the round-complexity of concurrent zero-knowledge. In Mihir Bellare, editor, *Advances in Cryptology—CRYPTO '00*, volume 1880 of *Lecture Notes in Computer Science*, pages 451–468. Springer-Verlag, August 2000.

[SV97] Amit Sahai and Salil P. Vadhan. A complete promise problem for statistical zero-knowledge. In *38th Annual Symposium on Foundations of Computer Science*, pages 448–457, Miami Beach, Florida, 20–22 October 1997. IEEE.

[Sch88] Uwe Schöning. Graph isomorphism is in the low hierarchy. *Journal of Computer and System Sciences*, 37(3):312–323, December 1988.

[Sha92] Adi Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, October 1992.

[Sip97] Michael Sipser. *Introduction to the Theory of Computation.* PWS Publishing, 1997.

[Vad99] Salil P. Vadhan. *A Study of Statistical Zero-Knowledge Proofs*. PhD thesis, Massachusetts Institute of Technology, August 1999.

[Vad00] Salil P. Vadhan. On transformations of interactive proofs that preserve the prover's complexity. In *Proceedings of the 32nd Annual ACM Symposium on Theory of Computing*, pages 200–207, Portland, OR, May 2000. ACM.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, 27–29 October 1986. IEEE.