Introduction to Complexity Theory*
Lecture 17: Zero-Knowledge Proof Systems

Notes taken by Michael Elkin and Ekaterina Sedletsky

May 19, 1999

Summary: In this lecture we introduce the notion of zero-knowledge interactive proof system, and consider an example of such a system (Graph Isomorphism). We define perfect, statistical and computational zero-knowledge and present a method for constructing zero-knowledge proofs for NP languages, which makes essential use of bit commitment schemes. Presenting a zero-knowledge proof system for an NP-complete language, we obtain zero-knowledge proof systems for every language in NP. We consider a zero-knowledge proof system for one NP-complete language, specifically Graph 3-Colorability. We mention that zero-knowledge is preserved under sequential composition, but is not preserved under the parallel repetition.

Oded's Note: *For alterantive presentations of this topic we refer the reader to either Section 2.3 in [2] (approx. 4 pages), or the 30-page paper [3], or the first 4–5 sections in Chapter 4 of [1] (over 50 pages).*

1 Definitions and Discussions

Zero-knowledge (ZK) is quite central to cryptography, but it is also interesting in context of this course of the complexity theory. Loosely speaking, zero-knowledge proof systems have the remarkable property of being convincing and yielding nothing beyond the validity of the assertion.

We say that the proof is zero-knowledge if the verifier does not get from it anything that he can not compute by himself, when it assumes that the assertion is true.

Traditional proof carries with it something which is beyond the original purpose. The purpose of the proof is to convince somebody, but typically the details of proof give the verifier *more* than merely conviction in the validity of the assertion and it is not clear whether it is essential or not.

But there is an extreme case, in which the prover gives the verifier nothing beyond being convinced that the assertion is true. If the verifier assumed a-priori that the assertion is true, then actually the prover supplied no new information.

The basic paradigm of zero-knowledge interactive proof system is that whatever can be efficiently obtained by interacting with a prover, could also be computed without interaction, just by assuming that the assertion is true and conducting some efficient computation.

Recall that in the definition of interactive proof system we have considered properties of the verifier, whereas no requirements on the prover were imposed. In zero-knowledge definition we talk about some *feature* of the prescribed prover, which captures prover's robustness against attempts

* Lecture Notes for a course given by Oded Goldreich at the Weizmann Institute of Science, Israel.

to gain knowledge by interacting with it. Verifier's properties are required to ensure that we have a proof system. A straightforward way of capturing the informal discussion follows.

Definition 1 *Let A and B be a pair of interactive Turing machines, and suppose that all possible interactions of A and B on each common input terminate in a finite number of steps. Then $\langle A, B\rangle(x)$ is the random variable representing the (local) output of B when interacting with machine A on common input x, when the random-input to each machine is uniformly and independently chosen.*

Definition 2 *Let (P, V) be an interactive proof system for some language L. We say that (P, V), actually P, is* zero-knowledge *if for every probabilistic polynomial time interactive machine V^* there exists an (*ordinary*) probabilistic polynomial time machine M^* so that for every $x \in L$ holds*

$$\{\langle P, V^*\rangle(x)\}_{x\in L} = \{M^*(x)\}_{x\in L},$$

where the equality "=" between the ensembles of distributions can be interpreted in one of three ways that we will discuss later.

Machine M^* is called a simulator *for the interaction of V^* with P.*

We stress that we require that *for every* V^* interacting with P, not merely for V, there exists a simulator M^*. This simulator, although not having access to the interactive machine P, is able to simulate the interaction of V^* with P. This fact is taken as evidence to the claim that V^* did not gain any knowledge from P (since the same output could have been generated without any access to P).

V^* is an interactive machine, potentially something more sophisticated than V, which is the prescribed verifier. What V^* is interested in, is to extract from the prover more information than the prover is willing to tell. The prover wants to convince the verifier in the fact that $x \in L$, but the verifier is interested to get *more* information. Any efficient way, which the verifier may try to do it, is captured by such interacting process or strategy V^*.

We are talking here about probability ensembles, but unlike our definitions of the pseudo-randomness, now probability ensembles are defined using index which is not a natural number, but rather less trivial. We have to modify the formalism slightly and to enable indexing by any countable set. It is important to understand that for every x we have two distributions $\langle P, V^*\rangle(x)$ and $M^*(x)$.

¿From now on the distribution ensembles are indexed by strings $x \in L$, so that each distribution contains only strings of length polynomial in the length of the index.

The question is when these two probability ensembles are equal or close.

There are three natural notions:

Definition 3 *Let (P, V) be an interactive proof system for some language L. We say that (P, V), actually P, is* perfect zero-knowledge(PZK) *if for every probabilistic polynomial time interactive machine V^* there exists an (*ordinary*) probabilistic polynomial time machine M^* so that for every $x \in L$ the distributions $\{\langle P, V^*\rangle(x)\}_{x\in L}$ and $\{M^*(x)\}_{x\in L}$ are identical, i.e.*

$$\{\langle P, V^*\rangle(x)\}_{x\in L} \equiv \{M^*(x)\}_{x\in L}.$$

We emphasize that a distribution may have several different ways of being generated. For example, consider the uniform distribution over n bits. The normal way of generating such a distribution would be to toss n coins and to write down their outcome. Another way of doing it would be to toss $2 * n$ coins, to ignore the coins in the even positions, and only to output the values of the coins in odd positions. These are two different ways of producing the same distribution.

Back to our definition, the two distributions $\{\langle P, V^* \rangle (x)\}_{x \in L}$ and $\{M^*(x)\}_{x \in L}$ have totally different ways of being produced. The first one is produced by interaction of the two machines and the second is produced by a traditional probabilistic polynomial time machine.

Consider a "honest" verifier. The two distributions are supposed to be exactly the same when x is in L. As we have seen in the example of uniform distribution, the two generated in different ways distributions may be identical, and our definition, indeed, requires them to be identical. The verifier has several other parameters, except of random coins and partial message history, and we do not fix them. The probability that the verifier accepts x, when $x \in L$, depends on these parameters. However, by completeness requirement of interactive proof system, this probability should be close to 1 for any possible values of the parameters.

When x is in L, M^* accepts with very high probability, but when x is not in L, it is not required that distribution $\{M^*(x)\}$ would be similar to $\{\langle P, V^* \rangle (x)\}$, and it may not happen. So machine M^*, which is called a simulator, simulates the interaction, assuming that x is in L. When x is not in L, it can do total rubbish.

To emphasize this point consider the following example.

Example 1.1: Consider a verifier V that satisfies the definition of interactive proof system, and so when $x \in L$, V will accept with very high (close to 1) probability. When $x \notin L$, V will accept with very low (close to 0) probability. Such $\langle P, V \rangle$ can be simulated by a trivial machine M which always accepts. When $x \in L$ the distributions $\langle P, V \rangle (x)$ and $M(x)$ are very close (the first is very close to 1 and the second is identically 1), whereas when $x \notin L$ the two distributions are totally different (the first one is close to 1 and the second one is identically 0). This behavior should not surprise, because if we would be able to simulate $\langle P, V \rangle$ by some non-interactive probabilistic polynomial time M it would follow that $IP \subseteq BPP$, whereas the definition we have introduced is much more general.

Example 1.2: Before we introduce an example of zero-knowledge interactive proof system, let us first describe an interactive proof system that "hides" some information from the verifier. Although, the system is not zero-knowledge, the verifier can not determine any particular bit of the "real" proof with probability bigger than 1/2.

Consider an NP relation R' that satisfies $(x, w') \in R$ if and only if $(x, \overline{w'}) \in R$. Such a relation can be created from any NP relation R by the following modification

$$R' \underline{\underline{\triangle}} \{(x, 0w) : (x, w) \in R\} \cup \{(x, 1w) : (x, \overline{w}) \in R\}.$$

Now we are suggesting the following interactive proof system by which the prover just sends to the verifier a witness w'.

$$\underline{Prover} \quad x \quad \underline{Verifier}$$

$$\xrightarrow[w']{\hspace{8em}}$$

Given the input x, the prover selects uniformly at random either w' or $\overline{w'}$, and sends one of them to the verifier. Both w' and $\overline{w'}$ are witnesses for x and so it is a valid interactive proof system. But if the verifier is interested to get some individual bit of the witness w', he has no way to do it using the data that he obtained from the prover. Indeed, each individual bit that the verifier receives from the prover is distributed uniformly and the verifier could produce such distribution by itself without interaction with the prover. However, we observe that the verifier gets some info about the witness, since it knows that he received either the witness or its complement.

Except of perfect zero-knowledge we define several more liberal notions of the same class. One of them requires that the distributions will be statistically close. By statistically close we mean that the variation distance between them is negligible as the function of the length of input x.

Definition 4 *The distribution ensembles $\{A_x\}_{x\in L}$ and $\{B_x\}_{x\in L}$ are* statistically close *or have* negligible variation distance *if for every polynomial $p(\cdot)$ there exists integer N such that for every $x \in L$ with $|x| \geq N$ holds*

$$\sum_{\alpha} |Prob\,[A_x = \alpha] - Prob\,[B_x = \alpha]| \leq \frac{1}{p(|x|)}.$$

Definition 5 *Let (P, V) be an interactive proof system for some language L. We say that (P, V), actually P, is* statistical zero-knowledge *proof system (SZK) or* almost perfect *interactive proof system if for every probabilistic polynomial time verifier V^* there exists non-interactive probabilistic polynomial time machine M^* such that the ensembles $\{\langle P, V^*\rangle(x)\}_{x\in L}$ and $\{M^*(x)\}_{x\in L}$ are statistically close.*

Even more liberal notion of zero-knowledge is computational zero-knowledge interactive proof system.

Definition 6 *Two ensembles $\{A_x\}_{x\in L}$ and $\{B_x\}_{x\in L}$ are computationally indistinguishable if for every probabilistic polynomial time distinguisher D and for every polynomial $p(\cdot)$ there exists integer N such that for every $x \in L$ with $|x| \geq N$ holds*

$$|Prob\,[D(x, A_x) = 1] - Prob\,[D(x, B_x) = 1]| \leq \frac{1}{p(|x|)}.$$

The probabilistic polynomial time distinguisher D is given an index of the distribution in the ensemble. This is a general notion of indistinguishability of ensembles indexed by strings.

Definition 7 *Let (P, V) be an interactive proof system for some language L. We say that (P, V), actually P, is computational* zero-knowledge *proof system (CZK) if for every probabilistic polynomial time verifier V^* there exists non-interactive probabilistic polynomial time machine M^* such that the ensembles $\{\langle P, V^*\rangle(x)\}_{x\in L}$ and $\{M^*(x)\}_{x\in L}$ are computationally indistinguishable.*

We should be careful about the order of quantifying in this definition. First we have to determine the verifier V^* and the simulating machine M^*, and then we should check whether the distributions are indistinguishable by "trying all possible" probabilistic polynomial time distinguishers D.

Typically, when we say zero-knowledge, then we mean computational zero-knowledge. This is the most liberal notion, but from the point of view of cryptography applications it is good enough, because, basically, it says that the non-interactive machine M^* can simulate the interactive system $\langle P, V^* \rangle$ in such a way, that "no one" can distinguish between them. Essentially, the generated distributions are close in the same sense as distribution generated by pseudo-random generator is close to the uniform distribution. The idea is that if the machine M^* is able to generate by itself, without interaction, the distribution, that is very close in computational sense to the distribution generated by V^*, that interacts with P, then V^* gains nothing from interaction with P.

Observe that zero-knowledge interactive proof system definition imposes three requirements. Completeness and soundness requirements follows from interactive proof system definition, and zero-knowledge definition imposes the additional condition.

The completeness condition fixes both the prover and the verifier, and states that when the both parties follow the prescribed protocol (both parties are "honest") and $x \in L$, then the verifier accepts with high probability. Observe that if either prover or verifier are "dishonest" the condition may not hold. Indeed, we can not quantify this condition over all verifiers, since some verifier may always reject and then of course the probability of accepting will be zero. On the other hand, if the prover is "dishonest", or in other words, there is another prover instead of P, then it may send some rubbish instead of witness in the NP case or instead of following the prescribed protocol in the general case, and the verifier will accept only with low probability, hence the condition will not hold.

The soundness condition fixes only the verifier and protects his interests. It says, that the verifier doesn't need to trust the prover to be honest. Soundness condition quantifies over all possible provers and says, that no matter how the prover behaves he has very small probability of convincing the verifier to accept a wrong statement. Of course, it is correct only for the fixed verifier V and not for a general one, since we may think about a verifier that always accepts. For such a verifier the probability to accept a wrong statement is 1, hence the soundness condition does not hold for him.

And the zero-knowledge condition protects the prover. It tells the prover: "You are entering the protocol which enables you to convince the other party, even if the other party does not trust you and does not believe you. You can be convinced, that you do not really give the other party more than you intended to (i.e. that the statement is true, nothing beyond that)."

The zero-knowledge condition fixes the prover and quantifies over all verifiers. It says that for any verifier, sophisticated and dishonest as much as he may be, he can not gain from the prover (which obeys the protocol, again bad prover may send all the information to the verifier and then, of course, zero-knowledge condition will not hold) more that the verifier could gain without interacting with the prover.

We finish the section by proving that BPP is contained in PZK. Indeed, any probabilistic polynomial time algorithm may be viewed as an interactive proof system without prover. In such a system no verifier can gain knowledge from the prover, since there is no prover. Thus the system is zero-knowledge. We formalize these considerations in the following claim.

Proposition 1.1 *$BPP \subseteq PZK$.*

Proof: Consider an interactive proof system in which the verifier V is just the probabilistic polynomial time algorithm, that decides the language L. Such V exists, since $L \in BPP$. The prover P is just a deterministic machine, that never sends data to the verifier. $< P, V >$ is an interactive proof system, since V will accept with probability $\geq \frac{2}{3}$, when $x \in L$, and will accept with probability $\leq \frac{1}{3}$, when $x \notin L$, since $L \in BPP$, and V decides L. Hence, the completeness and soundness conditions hold. Clearly, it is perfect zero-knowledge since for every V^*, the distribution that $< P, V^* >$ generates is identical to the distribution generated by V^* itself. ■

2 Graph Isomorphism is in Zero-Knowledge

Let $ISO \underline{\underline{\triangle}} \{(\langle G_1 \rangle, \langle G_2 \rangle) \mid G_1 \cong G_2\}$.

We assume that $ISO \notin BPP$, since otherwise $ISO \in ZK$, by Claim 1.1. So we are interested in showing zero-knowledge interactive proof systems for languages that are not in BPP, or at least are conjectured not to be in BPP.

Next, we introduce an interactive protocol proving that two graphs are isomorphic. The trivial interactive proof would be that the prover will send to the verifier the isomorphism, but this gives more information that the mere fact that the two graphs are isomorphic.

Instead of sending the isomorphism, which is not a good idea, we will use the following construction.

Construction 2.1 (Perfect Zero-Knowledge Proof for Graph Isomorphism)

- Common Input: A pair of two graphs, $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$. Let ϕ be an isomorphism between the input graphs, namely ϕ is a 1-1 and onto mapping of the vertex set V_1 to the vertex set V_2 so that $(u, v) \in E_1$ if only if $(\phi(u), \phi(v)) \in E_2$. Suppose that $|V_1| = |V_2| = n$ and the vertices of the both graphs are the numbers from 1 to n.

- Prover's first Step (P1): The prover selects a random isomorphic copy of G_2, and sends it to the verifier. Namely, the prover selects at random, with uniform probability distribution, a permutation π from the set of permutations over the vertex set V_1, and constructs a graph with vertex set V_1 and edge set

$$F \stackrel{def}{=} \{(\pi(u), \pi(v)) : (u, v) \in E_1\}$$

  The prover sends $H = (V_1, F)$ to the verifier.

- Motivating Remark: If the input graphs are isomorphic, as the prover claims, then the graph sent in step P1 is isomorphic to both input graphs. However, if the input graphs are not isomorphic then no graph can be isomorphic to both of them.

- Verifier's first Step (V1): Upon receiving a graph, $G' = (V', E')$, from the prover, the verifier asks the prover to show an isomorphism between G' and one of the input graph, chosen at random by the verifier. Namely, the verifier uniformly selects $\sigma \in \{1, 2\}$, and sends it to the prover (who is supposed to answer with an isomorphism between G_σ and G').

- Prover's second Step (P2): If the message, σ, received from the verifier equals 1 then the prover sends π to the verifier. Otherwise (i.e., $\sigma \neq 1$), the prover sends $\pi \circ \phi$ (i.e., the composition of π on ϕ) to the verifier. (Remark: the prover treats any $\sigma \neq 1$ as $\sigma = 2$.)

- Verifier's second Step (V2): If the message, denoted ψ, received from the prover is an isomorphism between G_σ and G' then the verifier outputs 1, otherwise it outputs 0.

For the schematic representation of the protocol, see the Diagram 2.1.

The verifier program presented above is easily implemented in probabilistic polynomial time. In case the prover is given an isomorphism between the input graphs as auxiliary input, also the prover's program can be implemented in probabilistic polynomial time. We now show that the above pair of interactive machines constitutes an almost perfect zero-knowledge interactive proof system for the language ISO.

Diagram 2.1

$\underline{Prover}$		$\underline{Verifier}$
$\pi \in_R Sym([n])$		
$H \longleftarrow \pi G_1$		
	$\xrightarrow[H]{}$	
		$\sigma \in_R \{1,2\}$
	$\xleftarrow[\sigma]{}$	
if $\sigma = 1$ then send $\psi = \pi$, otherwise $\psi = \pi \circ \phi^{-1}$		
	$\xrightarrow[\psi]{}$	
		Accept if and only if $H = \psi(G_\sigma)$

Theorem 8 *The construction above is an almost perfect zero-knowledge interactive proof system.*

Proof:

1. Completeness.

If the two graphs are isomorphic, we claim that the verifier will always accept. Indeed, if $\sigma = 1$, then the verifier comes to the end of the protocol with $H = \pi(G_1)$ and permutation $\psi = \pi$. The verifier checks whether

$$H = \psi(G_\sigma). \tag{1}$$

We observe that $\pi(G_1) = H$, and $\psi(G_\sigma) = \psi(G_1) = \pi(G_1)$, implying (2.1), hence the verifier accepts.

If $\sigma = 2$, then the verifier comes to the end of the protocol with $H = \pi(G_1)$ and permutation $\psi = \pi \circ \phi$. Again he verifies whether (1) holds. We observe that $\pi(G_1) = H$ and $\psi(G_\sigma) = \pi\phi(G_2) = \pi(G_1)$, again implying (1), hence the verifier accepts in both cases with probability 1.

The intuition is very simple. If the two graphs are isomorphic and the prover created the isomorphic copy to one of them, he should have no problem in showing isomorphism to each of the graphs.

2. Soundness.

Let $G_1 \ncong G_2$. Consider any prover P^*. If it sends to V a graph H, which is not isomorphic neither to G_1 nor to G_2, then this prover will have no way later to present an isomorphism from G_σ (no matter whether $\sigma = 1$ or 2) to H, *since there is no such isomorphism.* So, in this case, the probability of P^* to convince the verifier that $(\langle G_1 \rangle, \langle G_2 \rangle) \in ISO$, is zero. Suppose P^* sends an H that is isomorphic either to G_1 or to G_2. Without loss of generality, assume $H \cong G_1$. Then if the verifier randomly selected $\sigma = 1$, then P^* will be able to show the isomorphism between H and $G_\sigma = G_1$. Otherwise, if the verifier randomly selected $\sigma = 2$, then there is no isomorphism between $H \cong G_1$ and $G_\sigma = G_2$ (as otherwise G_1 and G_2 would be isomorphic), hence P^* will not be able to find one, despite his unlimited computational power. Hence, in this case, P^* will have probability of exactly $\frac{1}{2}$ to convince the verifier that $(\langle G_1 \rangle, \langle G_2 \rangle) \in ISO$ and we have shown that it is his optimum strategy. So

$$Prob[< P^*, V > (\langle G_1 \rangle, \langle G_2 \rangle) = accept \mid G_1 \ncong G_2] \leq \frac{1}{2}$$

for every prover P^*. By executing this protocol twice sequentially, we obtain

$$Prob[< P^*, V > (\langle G_1 \rangle, \langle G_2 \rangle) = accept \mid G_1 \ncong G_2] \leq \frac{1}{4},$$

hence, satisfying the soundness condition.

3. Zero-knowledge.

There is no other way to prove that the protocol is zero-knowledge, except of building a simulator and proving that it really generates the same distribution.

Simulator M^*. By definition of zero-knowledge we have to show that the distributions are the same when we are given an input from the language. On input $x \stackrel{def}{=} (\langle G_1 \rangle, \langle G_2 \rangle)$, simulator M^* proceeds as follows:

1. Setting the random-tape of V^*: *Let $q(\cdot)$ denote a polynomial bounding the running time of V^*. The simulator M^* starts by uniformly selecting a string $r \in \{0,1\}^{q(|x|)}$, to be used as the contents of the random-tape of V^*.*

2. Simulating the prover's first step (P1): *The simulator M^* selects at random,with uniform probability distribution, a "bit" $\tau = \{1, 2\}$ and a permutation ψ from the set of permutations over the vertex set V_τ. It then constructs an isomorphic copy G'' of the graph G_τ, i.e. $G'' = \psi(G_\tau)$.*

3. Simulating the verifier's first step (V1): *The simulator M^* initiates the execution of V^* by placing x on V^*'s common-input-tape, placing r (selected in step (1) above) on V^*'s random-tape, and placing G'' (constructed in step (2) above) on V^*'s incoming message-tape. After executing a polynomial number of steps of V^*, the simulator can read the outgoing message of V^*, denoted σ. Let us assume, without loss of generality, that the message sent by V^* is either 1 or 2. Indeed, if V^* sends $\sigma \notin \{1, 2\}$, then the prover has nothing to do with it and we may augment the prover P with a rule to ignore $\sigma \notin \{1, 2\}$ and just to wait for a "valid" σ. This would be very easy to simulate.*

4. Simulating the prover's second step (P2): *If $\sigma = \tau$ then the simulator halts with output (x, r, G'', ψ).*

5. Failure of the simulation: *Otherwise (i.e. $\sigma \neq \tau$), the simulator halts with output $\perp$.*

As could be seen from (4) we output the full view of the verifier. We stress that the simulator "has no way to know" whether V^* will ask to see an isomorphism to G_1 or to G_2.

This description of the simulator machine may confuse. Indeed, the definition of zero-knowledge considers the distributions of two random variables $\langle P, V^* \rangle (x)$ and $M^*(x)$, the outputs of $\langle P, V^* \rangle$ and M^* respectively. On the other hand, here M^* returns its whole view, that consists of all the data it possesses, specifically, x, r, G'' and ψ. This inconsistency can be treated by considering $\langle P, V^* \rangle (x)$ and $M^*(x)$ in the zero-knowledge definition as the views of V^* and M^* respectively, and by showing that this approach is equivalent to our definition.

Definition 9 *Let (P, V) be an interactive proof system for some language L. We say that (P, V), actually P, is* perfect zero-knowledge(PZK) by view *if for every probabilistic polynomial time interactive machine V^* there exists an (*ordinary*) probabilistic polynomial time machine M^* so that for every $x \in L$ holds*

$$\left\{ view_{(P,V^*)}(x) \right\}_{x \in L} \equiv \{ M^*(x) \}_{x \in L} \,,$$

where $view_{(P,V^*)}(x)$ is the final view of V^* after running $\langle P, V^* \rangle$ on input x and $M^*(x)$ is, as usual, the output of M^* after running on input x.

Claim 2.1 *An interactive proof system is perfect zero-knowledge if and only if it is perfect zero-knowledge by view.*

Proof: One direction is straightforward. Suppose there is a probabilistic polynomial time machine M^*, which for every input $x \in L$ outputs $M^*(x)$, that is distributed identically to the view of V^* at the end of execution of $\langle P, V^* \rangle$ on x. We observe that the last step of V^*, i.e. printing the output, is done without interaction with the prover. Note also that M^* may, instead of printing the output, write it down on its work-tape. Then M^* has on its work-tape the final view of V^*. Hence, it is capable to perform the last step of V^* and output the result and so the modified $M^*(x)$ is identical to $\langle P, V^* \rangle$, completing the proof of this direction.

In the opposite direction, we suppose that for every V^* there is a non-interactive probabilistic polynomial time machine M^*, which prints the same output, when it runs on x (for every $x \in L$), as V^* when $\langle P, V^* \rangle$ machine runs on x.

Consider some particular V^*. We need to show that there is a machine that for every $x \in L$ prints at the end of its execution on x the output identical to the view V^* at the end of execution of $\langle P, V^* \rangle$ on x. To see it, consider a verifier V^{**}, that behaves exactly like V^*, but outputs its whole view (i.e., it emulates V^* except that at the end it outputs the view of V^*). There is a machine M^{**}, such that its output $M^{**}(x)$ is distributed identically to the output of V^{**}, i.e. to the view of V^*. Thus M^{**} is the required machine. It completes the proof of the second direction, establishing the equivalency of the definitions. ■

Recall that the Definitions 3 and 9 both require that for every probabilistic polynomial time interactive machine V^* there exists an (*ordinary*) probabilistic polynomial time machine M^* with certain properties. Observe that in the proof of the non-trivial (i.e., second) direction of the above claim we use the fact that for every V^{**} (constructed out of V^*) there is a corresponding simulator. We stress that this was not used in the first direction in which we did not modify V^* (but rather M^*).

Statistical zero-knowledge by view and computational zero-knowledge by view are defined analogously. Similar claims about equivalency between statistical zero-knowledge and statistical zero-knowledge by view, and between computational zero-knowledge and computational zero-knowledge by view can be proved using the same argument as in the proof of Claim 2.1.

We claim that, when two graphs are isomorphic, then H gives no information on τ, because we can draw a correspondence between the possible mappings that can generate H from G_1 and the possible mappings that can generate H from G_2 by the isomorphism between the two graphs. It follows that

Claim 2.2 *Let $x = (\langle G_1\rangle, \langle G_2\rangle) \in ISO$. Then for every string r, graph H, and permutation ψ, it holds that*

$$Prob\left[view_{(P,V^*)}(x) = (x,r,H,\psi)\right] = Prob\left[M^*(x) = (x,r,H,\psi) \mid (M^*(x) \neq \perp)\right].$$

Proof: Let $m^*(x)$ describe $M^*(x)$ conditioned on its not being $\perp$. We first observe that both $m^*(x)$ and $\mathrm{view}_{(P,V^*)}(x)$ are distributed over quadruples of the form $(x,r,\cdot,\cdot)$, with uniformly distributed $r \in \{0,1\}^{q(|x|)}$, for some polynomial $q(\cdot)$. Let $v(x,r)$ be a random variable describing the last two elements of $\mathrm{view}_{(P,V^*)}(x)$ conditioned on the second element equals r. Similarly, let $\mu(x,r)$ describe the last two elements of $m^*(x)$ (conditioned on the second element equals r). Clearly, it suffices to show that $v(x,r)$ and $\mu(x,r)$ are identically distributed, for every x and r. Observe that once r is fixed the message σ sent by V^* on common input x, random-tape r, and incoming message H, is uniquely defined. Let us denote this message by $v^*(x,r,H)$. We need to show that both $v(x,r)$ and $\mu(x,r)$ are uniformly distributed over the set

$$C_{x,r} \stackrel{def}{=} \{(H,\psi) : H = \psi(G_{v^*(x,r,H)})\}.$$

The proof is slightly non-trivial because it relates (at least implicitly) to the automorphism group of the graph G_2 (i.e., the set of permutations π for which $\pi(G_2)$ is identical, not just isomorphic, to G_2). For simplicity, consider the special case in which the automorphism group of G_2 consists of merely the identity permutation (i.e., $G_2{=}\pi(G_2)$ if and only if π is the identity permutation). In this case, $(H,\psi) \in C_{x,r}$ if and only if H is isomorphic to both G_1 and G_2 and ψ is the isomorphism between H and $G_{v^*(x,r,H)}$. Hence, $C_{x,r}$ contains exactly $|V|!$ pairs, each containing a different graph H as the first element, proving the claim in the special case.

For the proof of the general case we refer the reader to [3] (or to [1]). ■

Recall that to prove perfect zero-knowledge we need to show that $\mathrm{view}_{(P,V^*)}(x)$ and $M^*(x)$ are identically distributed. Here, it is not the case. Although, $\mathrm{view}_{(P,V^*)}(x)$ and $M^*(x)|(M^*(x) \neq\perp)$ are identically distributed, when $M^*(x) =\perp$ the distributions are totally different. A common way to overcome this difficulty is to change the definition of perfect zero-knowledge, at least a bit. Suppose we allow the simulator to output a special symbol which we call "failure", but we require that this special symbol is outputted with probability at most 1/2. In this case the construction would satisfy the definition and we could conclude that it is a perfect zero-knowledge proof (under the changed definition).

But recall that Theorem 8 states that the construction is almost perfect zero-knowledge. This is indeed true, without any change of the definitions, if the simulator reruns steps (2)-(4) of the construction $|x|$ times. If, at least once, at step (4) σ is equal to τ, then output (x,r,G'',ψ). If at all $|x|$ trials $\sigma \neq \tau$, then output rubbish. In such a case the simulation will not be perfect, but will be statistically close, because the statistical difference will be $2^{-|x|}$.

It remains to show that the running time of the simulator is polynomial in $|x|$. In the case when we run it $|x|$ times, it is obvious, concluding our proof that the interactive proof system is almost perfect zero-knowledge. In the case when we change the definition of perfect zero-knowledge in the described above way, we are done by one iteration, hence the running time is against polynomial

Another possibility is to allow the simulator to run expected polynomial time, rather than strict polynomial time, in such a case the interpretation would be to rerun steps (2)-(4), until the output is not $\perp$. Every time we try, we have a success probability of exactly 1/2. Hence, the expected number of trials is 2.

This concludes the proof of the Theorem 8. ■

These definitions, one allowing the failure probability and another allowing an expected polynomial time, are not known to be equivalent. Certainly, if we have a simulator which with probability at most 1/2 outputs the failure, then it can be always converted to one which runs in expected polynomial time. But the opposite direction is not known and is not clear.

3 Zero-Knowledge Proofs for NP

3.1 Zero-Knowledge NP-proof systems

We want to show why it was essential to introduce the interactive proofs in order to discuss zero-knowledge (in a non-trivial way). One can also define zero-knowledge for NP-proofs, but by the following claim such proofs exist only for BPP (and are thus "useless").

Proposition 3.1 *Let L be a language that admits zero-knowledge NP-proof system. Then $L \in BPP$.*

Proof: For the purpose of the proof we use only the fact that an "honest" verifier V, which outputs its whole view, can be simulated by a probabilistic polynomial time machine M.

(One may think that in view of Claim 2.1 there is no point to specify that V outputs its whole view. It is not correct. In the proof of the claim we used the fact that if an interactive proof system is zero-knowledge, then *for any* verifier V^* there is a simulator M^* with certain properties. Once we fix a verifier, such kind of argument no longer works. It is the reason that we specify the output of V.)

Let L be the language that the NP proof system decides and R_L its NP relation. Let $x \in L$. The view of the verifier when interacting with a prover on input x will be the input itself and the message, call it w, that he received. Since we consider an honest prover, the following holds

$$view_{(V,P)}(x) = (x, w) \in R_L.$$

Simulator M simulates this view on input x. We show that $(x, M(x)) \in R_L$ with high probability, specifically

$$Prob[(x, M(x)) \in R_L] \geq \frac{2}{3}. \tag{2}$$

Also, we will show that for $x \notin L$

$$Prob[(x, M(x)) \in R_L] = 0 < \frac{1}{3}, \tag{3}$$

hence, M is the probabilistic polynomial time algorithm for L, implying $L \in BPP$.

First, discuss the case of $x \in L$ and suppose for contradiction that

$$Prob[(x, M(x)) \in R_L] < \frac{2}{3}. \tag{4}$$

Then we claim that there is a deterministic polynomial time distinguisher D, that distinguishes between the two distributions $\langle P, V \rangle (x)$ and $M(x)$ with non-negligible probability. Consider $D(\alpha)$ which is defined as 1 if $\alpha \in R_L$, and 0 otherwise, where α is a pair of the form (x, w). Obviously, for $x \in L$,

$$Prob[D(x, \langle P, V \rangle (x)) = 1] = 1$$

since P is a "honest" prover, i.e. a prover that supplies witness for $x \in L$.

Also, from (4) follows that

$$Prob[D(x, M(x)) = 1] < \frac{2}{3}.$$

So there is a non-negligible gap between the two cases. It contradicts the assumption, that the distribution of $M(x)$ is polynomially indistinguishable from the distribution of $\langle P, V \rangle (x)$.

On the other hand, when $x \notin L$, then by the definition of NP, there is no witness y, such that $(x, y) \in R_L$. Particularly, $(x, M(x)) \notin R_L$. In other words,

$$Prob[(x, M(x)) \in R_L] = 0.$$

Concluding the proof of (2) and (3), hence $L \in BPP$. ∎

3.2 NP $\subseteq$ ZK (overview)

Now, we are going to show that NP has a zero-knowledge interactive proof ($NP \subseteq ZK$).

We will do it assuming, that we have some magic boxes, called commitment schemes, and later we will describe their implementation.

Commitment schemes are used to enable a party to commit itself to a value while keeping it secret. In a latter stage the commitment is "opened" and it is guaranteed that the "opening" can yield only a single value determined in the committing phase. Commitment schemes are the digital analogue of non-transparent sealed envelopes. Nobody can look inside the envelopes and know the value. By putting a note in such an envelope a party commits itself to the contents of the note while keeping it secret.

We present a zero-knowledge proof system for one NP-complete language, specifically *Graph 3-Coloring.*

The language *Graph 3-Coloring*, denoted $G3C$, consists of all simple graphs (i.e., no parallel edges or self-loops) that can be *vertex-colored* using 3 colors so that no two adjacent vertices are given the same color. Formally, a graph $G = (V, E)$, is $3-colorable$, if there exists a mapping $\phi : V \longmapsto \{1, 2, 3\}$, so that $\phi(u) \neq \phi(v)$ for every $(u, v) \in E$.

In general, if we want to build a zero-knowledge interactive proof system for some other NP language, we may just use standard reduction and run our protocol on the reduced instance of the graph colorability. Thus, if we can show a zero-knowledge proof for one $NP-$complete language, then we can show for all. Basically it is correct, although inaccurate. For more details see [3].

One non-zero-knowledge proof is to send the coloring to the verifier, which would check it, and this would be a valid interactive proof system, but, of course, not a zero-knowledge one.

The instructions that we give to the prover can actually be implemented in polynomial time, if we give the prover some auxiliary input, specifically the 3-coloring of the graph, which is the NP-witness that the graph is in $G3C$. Let ψ be a 3-coloring of G. The prover selects at random some permutation. But this time it is not a permutation of the vertices, but rather permutation of the colors

$$\pi \in_R Sym([3]).$$

Then it sets $\phi(v) \stackrel{\triangle}{=} \pi(\psi(v))$, for each $v \in V$ and puts each of these permuted colors in a separate locked box (of the type that was discussed before), and the boxes have marks, so that both the prover and verifier know the number of each box.

$$\boxed{\phi(1)}^1, \boxed{\phi(2)}^2, \ldots\ldots, \boxed{\phi(i)}^i, \ldots\ldots, \boxed{\phi(n)}^n$$

The ϕ-color of vertex i is sent in the box number i. The verifier can not see the contents of the boxes, but the prover claims, that he has sent a legal coloring of the graph, which is a permutation of the original one.

The verifier selects an edge $e = (u, v)$ at random, and sends it to the prover.

$$\underline{P} \xleftarrow[\quad e = (u,v) \qquad e \in_R Edge(G) \quad]{} \underline{V}$$

Basically, it asks to inspect the colors of vertices u and v. It expects to see two different colors, and if they are not, then the verifier knows that something is wrong.

The prover sends back the key to box number u and the key to box number v.

$$\underline{P} \xrightarrow[\quad key_u \quad and \quad key_v \quad]{} \underline{V}$$

The verifier uses the keys to open these two boxes (other boxes remain locked), and looks inside. If he finds two different colors from the set $\{1, 2, 3\}$, then it accepts. Otherwise, he rejects. This completes the description of interactive proof system for $G3C$, that we call $G3C$ protocol.

Even more drastically then before, this will be a very weak interactive proof. In order to make any sense we have to repeat it a lot of times. Every time we repeat it, a prover selects a new permutation π, so crucially the color that the verifier sees in one iteration have nothing to do with the colors, that he sees in other iterations.

Of course, if the prover would always use the same colors, then the verifier would know the original coloring by just asking sufficient number of times, a different edge each time. So it is crucial, that in every iteration the coloring is totally random. We observe that the randomness of the coloring may follow also from a random choice of the original coloring ϕ and not only from the randomness of the permutation π, which has very small sample space (only 3!=6). A computationally unbounded prover will have no problem with randomly selecting a 3-coloring for a graph. On the other hand, a polynomial time bounded prover has to be feeded with all the 3-colorings as auxiliary inputs, in order to be able to select randomly at uniform an original coloring ϕ.

Observe that zero-knowledge property (when augmenting the definition a bit) is preserved under sequential composition. The error probabilities also decrease if we apply the protocol in parallel. But in general, it is not known that zero-knowledge is preserved under parallel composition. In particular, the protocol which is derived by running $G3C$ protocol in parallel many times is in some sense not zero-knowledge, or at least probably is not zero-knowledge.

Proposition 3.2 *$G3C$ protocol is zero-knowledge interactive proof system.*

Proof:

1. Completeness.

If the graph is 3-Colorable, and both the prover and the verifier follow the protocol, then the verifier will always accept. Indeed, since ϕ is a legal coloring, for every edge $e = (u, v)$ holds $\phi(u) \neq \phi(v)$. Hence, for $G \in G3C$

$$Prob[\langle P, V\rangle (\langle G\rangle) = accept] = 1,$$

and we are done.

2. Soundness.

Let ϕ be a color-assignment of graph G that the prover uses in his trial to convince the verifier that the graph is 3-colorable. If $G \notin G3C$, then by definition of 3-Colorability there exists an edge

$$e_0 = (u, v) \in Edges(G),$$

such that either $\phi(u) = \phi(v)$ or $\phi(u) \notin \{1, 2, 3\}$. Without loss of generality, suppose $\phi(x) \in \{1, 2, 3\}$ for all x's. If the verifier asked to open e_0 then by commitment property (specifically, by unambiguity requirement of a commitment scheme, to be discuss it in the next subsection) he will reveal that $\phi(u) = \phi(v)$ and thus he will reject. I.e.

$$\text{Prob}\left[\begin{array}{c}\text{a randomly selected edge}\\ e = (w, z) \text{ satisfies } \phi(w) \neq \phi(z)\end{array}\right] \leq \frac{|Edges(G)\backslash\{e_0\}|}{|Edges(G)|} = 1 - \frac{1}{|Edges(G)|}.$$

Hence, for any prover P^*, for $G \notin G3C$

$$Prob[\langle P^*, V\rangle (G) = accept] \leq 1 - \frac{1}{|Edges(G)|}.$$

By repeating the protocol sequentially, sufficiently many times, specifically $\lceil \log_{\left(\frac{|Edges(G)-1|}{|Edges(G)|}\right)} \frac{1}{3} \rceil$, we reduce this probability to

$$Prob[\langle P^*, V\rangle (G) = accept|\ G \notin G3C] < \frac{1}{3}$$

for any prover P^*.

3. Zero-Knowledge.

In order to show that $G3C$ protocol is a zero-knowledge proof system, we devise for every verifier V^* the following simulating non-interactive machine M^*, that for every $G \in G3C$ generates a distribution $m^*(\langle G \rangle) = M^*(\langle G \rangle) | (M^*(G) \neq \perp)$ identical to the distribution $\langle P, V^* \rangle (\langle G \rangle)$.

Description of M^*
(1) Fix the random coins r of V^*.
(2) Select at random an edge $e' = (u', v')$,

$$e' = (u', v') \in_R Edges(G).$$

(3) V^* sends to the M^* (in fact, it is inter-routine communication, since V^* is incorporated in M^*) boxes that are filled in the following way. All the boxes are filled with garbage, except the two boxes of u' and v', which are filled with two different, randomly selected numbers between 1 and 3, namely

$$c \neq d \in_R \{1, 2, 3\}.$$

Put c in u' box, d in v' box.

$$\boxed{} \;\cdots\cdots\cdots\; \underset{\substack{\uparrow \\ u'}}{\boxed{\;\;c\;\;}} \;\cdots\cdots\; \underset{\substack{\uparrow \\ v'}}{\boxed{\;\;d\;\;}} \;\cdots\cdots\cdots\; \boxed{}$$

(4) If V^* chooses e', M^* sends V^* the keys and the simulation is completed. If the verifier sends a different edge, we rerun everything from step (1).

Totally, rerun everything at most $|Edges(G)|$ times. If at least once V^* selected e' in that iteration, print the output of V^*. Otherwise, output $\perp$.

V^* has no way to know what boxes are filled with garbage and what boxes are filled with something meaningful, since the boxes are "non-transparent" (see below about secrecy requirement of a commitment scheme). Thus the probability, that in each iteration V^* will select e' is $\frac{1}{|Edges(G)|}$. By doing $|Edges(G)|$ iterations we reach a constant probability (of about e^{-1}) of generating distribution exactly identical to $\langle P, V^* \rangle (\langle G \rangle)$.

Consider the distribution of

$$m^*(\langle G \rangle) = M^*(\langle G \rangle) \mid (M^*(\langle G \rangle) \neq \perp).$$

For every $G \in G3C$, if M^* did not output $\perp$, then one of the iterations was successful. In this case V^* has selected e', which is a randomly selected edge from the graph G. M^* prints the output of V^* on e'. Thus, the distribution of $m^*(\langle G \rangle)$ is identical to the distribution of the output of V^* when it is given a randomly selected edge from the graph G. Now consider the distribution of $\langle P, V^* \rangle (\langle G \rangle)$. The prescribed prover P is fixed and by the construction it supplies to V^* a randomly selected edge from the graph G. Thus the distribution of $\langle P, V^* \rangle (\langle G \rangle)$ is also identical to the distribution of the output of V^* when it is given a randomly selected edge from the graph. Hence,

$$m^*(\langle G \rangle) = \langle P, V^* \rangle (\langle G \rangle).$$

So if the boxes are perfectly sealed, then $G3C \in PZK$. ■

3.3 Digital implementation

In reality we use *commitment scheme* instead of these "sealed envelopes", that we discussed previously. In general, a *commitment scheme* is an efficient *two-phase* two-party protocol through which one party, called the *sender* (S), can commit itself to a *value* so the following two conflicting requirements are satisfied.

1. *Secrecy*: At the end of the first phase, the other party, called the *receiver* (R), does not gain any knowledge of the sender's value. This requirement has to be satisfied for any polynomial-time receiver.

2. *Unambiguity*: Given the transcript of the interaction in the first phase, there exists at most one value which the receiver may later (i.e., in the second phase) accept as legal "opening" of the commitment. This requirement has to be satisfied even if the sender tries to cheat (no matter what strategy it employs).

In addition, we require that the protocol is *viable* in the sense that if both parties follow it then, at the end of the second phase, the receiver gets the value committed to by the sender.

Denote by $S(s, \sigma)$ the message that the sender sends to the receiver when he wants to commit himself to a bit σ and his random coins are s. The secrecy requirement states that, for random s, the distributions of the random variables $S(s, 0)$ and $S(s, 1)$ are indistinguishable by polynomial-size circuits.

Let view of S (R), denoted as $View(S)$ ($View(R)$), be the collection of all the information known to the sender (receiver). Denote by r the random coins of R and let $\overline{m}$ be the sequence of messages that R received from S. Then $View(R) = (r, \overline{m})$. In case of single-round commit, $\overline{m} = S(s, \sigma)$. When the sender S wants to commit itself to a bit σ and his random coins sequence is s, then $View(S) = (s, \sigma)$. The unambiguity requirement states that for all but a negligible fraction of r's, there is no such $\overline{m}$ for which there exist two sequences of random coin tosses of S, s and s' such that

$$View(S) = (s, 0) \;\; and \;\; View(R) = (r, \overline{m})$$

and

$$View(S) = (s', 1) \;\; and \;\; View(R) = (r, \overline{m}).$$

The intuition of this formalism is that if such s and s' would exist, then the receiver's view would not be monosemantic. Instead, it would enable to sophisticated sender to claim that he committed either to zero or to one, and the receiver would not be able to prove that the sender is cheating.

If we think about the analogy of sealed envelopes, we send them and we believe that their contents is already determined. However, if we do not open them, they look the same whatever the contents is.

In the rest of this section we will need commitment schemes with a seemingly stronger secrecy requirement that defined above. Specifically, instead of requiring secrecy with respect to all polynomial-time machines, we will require secrecy with respect to all (not necessarily uniform) families of polynomial-size circuits. Assuming the existence of non-uniformly one-way functions commitment schemes with nonuniform secrecy can be constructed, following the same constructions used in the uniform case.

Proposition 3.3 *The interactive proof system for G3C that uses bit commitment schemes instead of the "magic boxes" for sending colors is still zero-knowledge.*

Proof:

1. Completeness.

If the graph is 3-Colorable, the prover (the sender) will have no problem to convince the verifier (the receiver) by sending the "right keys" to the commitment schemes, that contain the colors of the endpoints of the edge that the verifier asked to inspect. More formally, by sending the "right keys" we mean performing the reveal phase of the commitment scheme. It takes the following form:

1. The prover sends to the verifier the bit σ (the contents of the sealed envelope) and its random coins, s, that he has used when he committed to the bit (we call it the *commit phase*).

2. The verifier checks that σ and s and his own random coins r indeed yield messages that the verifier has received in the commit phase. This verification is done by running the segment of the prover's program in which it committed to the bit σ and the verifier's program. Both programs are now run by the verifier with fixed coins and take polynomial time. Observe that the verifier could not run the whole prover's program, since it need not to be polynomial. As previously, the probability of accepting a 3-Colorable graph is 1.

2. Soundness.

The unambiguity requirement of the commitment scheme definition ensures that the soundness is satisfied too. We will have a very small increase in the probability to accept a non-3-Colorable graph, relatively to the case when we used magic boxes. As in the proof of Claim 3.2 let G be a non-3-Colorable graph and ϕ the assignment of colors that the prover uses. If the interactive proof system is based on magic boxes, then the probability to accept G is exactly equal to the probability of selecting a properly colored by ϕ edge from G while selecting one edge uniformly at random. As we have seen in the proof of Claim 3.2, this probability (further denoted p_0) is bounded by $\frac{|Edges(G)|-1}{|Edges(G)|}$. Intuitively, p_0 is the probability that the verifier asked to inspect an edge that is properly colored by ϕ, although ϕ is not a proper coloring.

If the proof system is based on commitment schemes rather than on magic boxes, then except of p_0 there is a probability that the verifier asked to inspect a non-properly colored edge, but the prover has succeeded to cheat him. It may happen only if the verifier's random coins r belong to the fraction of all possible random coins of the verifier, for which there are random coins of the prover, which enable him to pretend that he committed both to 0 and to 1. Unambiguity requirement of the commitment scheme ensures that this fraction is negligible, and hence the probability (further denoted p_1) that r belongs to this fraction is negligible too. Thus we can bound p_1 by $\frac{1}{2*|Edges(G)|}$. So, the total probability to accept a non-3-Colorable graph is bounded by

$$p_0 + p_1 \leq \frac{|Edges(G)|-1}{|Edges(G)|} + \frac{1}{2*|Edges(G)|} = \frac{|Edges(G)|-1/2}{|Edges(G)|}.$$

By repeating the protocol sufficiently many times we can make this probability smaller than $\frac{1}{3}$, thus satisfying the soundness property.

3. Zero-Knowledge.

The zero-knowledge, that will be guaranteed now, is a computational zero-knowledge.

To show this we prove that M^* outputs $\perp$ with probability at most $\frac{1}{2}$, and that, conditioned on not outputting $\perp$, the simulator's output is computationally indistinguishable from the verifier's view in a "real interaction with the prover".

Claim 3.4 *For every sufficiently large graph, $G = (V, E)$, the probability that $M^*(\langle G\rangle) = \perp$ is bounded above by $\frac{1}{2}$.*

Proof: As above, n will denote the cardinality of the vertex set of G. Let $e_1, e_2, ..., e_n$ be the contents of the n "sealed envelopes" that the prover/the simulator sends to the verifier. Let $s_1, s_2, ..., s_n$ be the random coins that the prover used in the commit phase while committing to $e_1, e_2, ..., e_n$. Let us denote by $p_{u,v}(G, r, (e_1, e_2, ..., e_n))$ the probability, taken over all the choices of the $s_1, s_2, ..., s_n \in \{0,1\}^n$, that V^*, on input G, random coins r, and prover message $(C_{s_1}(e_1), ..., C_{s_n}(e_n))$, replies with the message (u, v). We assume, for simplicity, that V^* always answers with an edge of G (since otherwise its message is anyhow treated as if it were an edge of G). We claim that for every sufficiently large graph, $G = (V, E)$, every $r \in \{0,1\}^{q(n)}$, every edge $(u, v) \in E$, and every two sequences $\alpha, \beta \in \{1,2,3\}^n$, it holds that

$$|p_{u,v}(G, r, \alpha) - p_{u,v}(G, r, \beta)| \leq \frac{1}{2|E|}$$

This is proven using the non-uniform secrecy of the commitment scheme.
For further details we refer the reader to [3] (or to [1]). ■

Claim 3.5 *The ensemble consisting of the output of M^* on input $G = (V, E) \in G3C$, conditioned on it not being $\perp$, is computationally indistinguishable from the ensemble $view_{(P,V^*)}(\langle G\rangle)_{G\in G3C}$. Namely, for every probabilistic polynomial-time algorithm, A, every polynomial $p(\cdot)$, and every sufficiently large graph $G = (V, E)$,*

$$\left|Pr(A(m^*(\langle G\rangle)) = 1) - Pr(A(view_{(P,V^*)}(\langle G\rangle)) = 1)\right| < \frac{1}{p(|V|)}$$

We stress that these ensembles are very different (i.e., the statistical distance between them is very close to the maximum possible), and yet they are computationally indistinguishable. Actually, we can prove that these ensembles are indistinguishable also by (non-uniform) families of polynomial-size circuits. In first glance it seems that Claim 3.5 follows easily from the secrecy property of the commitment scheme. Indeed, Claim 3.5 is proven using the secrecy property of the commitment scheme, yet the proof is more complex than one anticipates at first glance. The difficulty lies in the fact that the above ensembles consist not only of commitments to values, but also of an opening of some of the values. Furthermore, the choice of which commitments are to be opened depends on the entire sequence of commitments.
Proof: The proof can be found in [3] (or to [1]). ■

This completes the proof of computational zero-knowledge and of Proposition 3.3. ■

Such a bit commitment scheme is not difficult to implement. We observe that the size of the intersection between the space of commitments to 0 and the space of commitments to 1 is a negligible fraction of the size of the union of the two spaces, since a larger intersection would violate the unambiguity requirement. Hence, the space of commitments to 0 and the space of commitments to 1 almost do not intersect. On the other hand, commitments to 0 should be indistinguishable from commitments to 1. Using mechanisms of one-way functions and hard-core bits, we can satisfy the seemingly conflicting requirements.

Consider the following construction.

Let $f : \{0,1\}^n \longrightarrow \{0,1\}^n$ be one-way permutation, and $b : \{0,1\}^n \longrightarrow \{0,1\}$ be its hard-core bit, where n is a security parameter. To commit itself to a bit $v \in \{0,1\}$, the sender S selects uniformly at random $s \in_R \{0,1\}^n$ and sends $(f(s), b(s) \oplus v)$ to the receiver R. R stores this pair as $\alpha = f(s)$ and $\sigma = b(s) \oplus v$.

In the second phase (i.e. when sending the "keys"), S reveals its random coins s. R calculates $v = \sigma \oplus b(s)$, and accepts v if $\alpha = f(s)$. Otherwise, R rejects, since if $\alpha \neq f(s)$, then the sender tries to cheat.

Proposition 3.6 *The protocol is a bit commitment scheme.*

Proof:

Secrecy: For every receiver R^* consider the distribution ensembles $\langle S(0), R^* \rangle (1^n)$ and $\langle S(1), R^* \rangle (1^n)$. Observe that

$$\langle S(0), R^* \rangle (1^n) = (f(s), b(s))$$

and

$$\langle S(1), R^* \rangle (1^n) = (f(s), \overline{b(s)}).$$

By definition of hard-core bit $b(\cdot)$ of a one-way function $f(\cdot)$, for every probabilistic polynomial time algorithm A, every polynomial $p(\cdot)$ and for every sufficiently large random string s

$$Pr[A(f(s)) = b(s)] < \frac{1}{2} + \frac{1}{p(|s|)}.$$

In other words, the bit $b(s)$ is *unpredictable* by probabilistic polynomial time algorithm given $f(s)$. Thus the distributions $(f(s), b(s))$ and $(f(s), \overline{b(s)})$ are probabilistic polynomial time indistinguishable. (For the proof of equivalency between indistinguishability and unpredictability see the previous lecture.)

Hence for any probabilistic polynomial time distinguisher D

$$\left| Prob[D(f(s), b(s)) = 1] - Prob[D(f(s), \overline{b(s)} = 1]) \right| < \frac{1}{p(|s|)}$$

proving the secrecy.

Unambiguity: We claim that there is no r for which $(r, \overline{m})$ is ambiguous, where $\overline{m}$ is the sequence of messages that S sent to R. Suppose,that $(r, \overline{m})$ is a possible 0-commitment, i.e. there exist a string s such that, $\overline{m}$ describes the messages received by R, where R uses local coins r and interacts with S, which uses local coins s and has input $v = 0$ and security parameter n. Also, suppose for contradiction that $(r, \overline{m})$ is a possible 1-commitment.

Then there exists s_1 such that $View(S) = (s_1, 0, 1^n)$, $View(R) = (r, \overline{m})$.

And there exists s_2 such that $View(S) = (s_2, 1, 1^n)$, $View(R) = (r, \overline{m})$.

But then $\overline{m}$=$(f(s_1), b(s_1))$=$(f(s_2), \overline{b(s_2)})$. I.e. $f(s_1) = f(s_2)$, implying $s_1 = s_2$ since $f(\cdot)$ is a permutation. But then $b(s_1) = b(s_2)$, contradicting $b(s_1) = \overline{b(s_2)}$. Hence, the assumption that there exists ambiguous $(r, \overline{m})$ leads to the contradiction. Therefore, the unambiguity requirement is satisfied, implying that the protocol is a one-bit commitment scheme. ∎

4 Various comments

4.1 Remark about parallel repetition

Recall that the definition of zero-knowledge requires that for every V^* there exist a probabilistic polynomial time simulator M^* such that for every $x \in L$ the distributions of $\langle P, V^* \rangle (x)$, $M^*(x)$ are identical/statistically close/computationally indistinguishable.

A particular case of this concept is a blackbox zero-knowledge.

Definition 10 (Blackbox Zero-Knowledge): *Let (P, V) be an interactive proof system for some language L. We say that (P, V) is* perfect/statistical/computational blackbox zero-knowledge *if there exists an oracle machine M, such that for every probabilistic polynomial time verifier V^* and for every $x \in L$, the distributions of $\langle P, V^* \rangle (x)$, $M^{V^*}(x)$ are identical/statistically close/computationally indistinguishable.*

Recall that M^{V^*} is an oracle Turing machine M with access to oracle V^*. The following theorem is given without proof.

Theorem 11 *If there is an interactive proof system (P, V)* (with negligible error probability) *for language L that satisfies*

- *The prescribed verifier V, sends the outcome of each coin it tosses (i.e. the interactive proof system is of public-coin type).*
- *The interactive proof system consists of constant number of rounds.*
- *The protocol is blackbox zero-knowledge.*

Then $L \in BPP$.

Observe that $G3C$ protocol is blackbox zero-knowledge, but its failure probability is $1 - \frac{1}{|E|}$, hence it is not negligible. Blackbox zero-knowledge is preserved under sequential composition. Thus by repeating $G3C$ polynomially many times we obtain a blackbox zero-knowledge protocol with negligible error probability, but the number of rounds of this protocol is no more constant, thus violating the second condition of the theorem.

Blackbox zero-knowledge is not preserved under parallel composition. Indeed, there exist zero-knowledge proofs that when repeated twice in parallel do yield knowledge. Repeating G3C protocol polynomially many times in parallel clearly satisfies all conditions of the Theorem 11 except blackbox zero-knowledge. Thus, unless $NP \subseteq BPP$, this parallel protocol is not black-box zero-knowledge.

More generally, our unability to construct an interactive proof system that satisfies all the conditions of Theorem 11 $G3C$ does not surprise, since if we could construct such an interactive proof system, then by Theorem 11, $G3C$ would belong to BPP and since $G3C$ is NP-complete it would imply $NP \subseteq BPP$.

> Oded's Note: *All known zero-knowledge proof systems are proven to be so via a black-box argument, and it seems hard to conceive an alternative. Thus practially speaking, Theorem 11 may be understood as saying that zero-knowledge proofs (with negligible error) for languages outside BPP should either use non-constant number of rounds or use "private coins" (i.e., not be of public-coin type).*

4.2 Remark about randomness in zero-knowledge proofs

In interactive proofs it was important that the verifier took random steps. But as we mentioned, the prover could be deterministic. The only advantage of probabilistic prover in an interactive proof system is that it may be more efficient.

On the other hand, the prover in a zero-knowledge proof system has to be randomized, not because it has not enough power to do things deterministically, but rather because a deterministic prover will not be able to satisfy the zero-knowledge requirement. In $G3C$ example, suppose that the prover selected the permutation π in a very complicated and secret way, but deterministically. Then a simple verifier will just exhaust all the edges, when repeating the protocol several times. Hence, such a protocol would not be a zero-knowledge. In general, we may prove that if a language has a zero-knowledge proof in which either prover or verifier is deterministic, then the language is in BPP.

5 Bibliographic Notes

For the comprehensive discussion of zero-knowledge and commitment scheme see Chapter 4 in [1]. Theorem 11 appear in [4]

1. Oded Goldreich, *Foundations of Cryptography – fragments of a book.* February 1995. Revised version, January 1998. Both versions are available from `http://theory.lcs.mit.edu/~oded/frag.html`.

2. Oded Goldreich, *Modern Cryptography, Probabilistic proofs and Pseudorandomness*, Springer Verlag, Berlin, 1998.

3. O. Goldreich, S. Micali, A. Wigderson, *Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems.* *JACM*, Vol. 38, No. 1, pages 691–729, 1991. Preliminary version in *27th FOCS*, 1986.

4. O. Goldreich and H. Krawczyk, *On the Composition of Zero-Knowledge Proof Systems*, *SIAM Journal on Computing*, Vol.25, No. 1, February 1996, pages 169-192.