6.876/18.426: Advanced Cryptography February 24, 2003

Lecture 5: ZK proofs for all of NP

Scribed by: Dah-Yoh Lim

1 Recap

Recall that (P, V) is a ZKPS for a language L iff the following conditions hold:

1. **Completeness:** $\forall x \in L,\ Pr[(\mathsf{P}, \mathsf{V})[x] = \text{“}Yes\text{”}] \geq 1 - negl(k)$.
2. **Soundness:** $\forall x \notin L, \forall \mathsf{P}'\ Pr[(\mathsf{P}', \mathsf{V})[x] = \text{“}Yes\text{”}] \leq 1/2$.
3. **Zero Knowledgeness:**
$\forall \mathsf{V}'_{\mathsf{PPT}}, \exists \mathsf{S}_{\mathsf{PPT}}, \forall x \in L, \forall a \in \{0,1\}^{poly(|x|)}, VIEW_{\mathsf{V}'}^{\mathsf{PV}'}(x; x, a) \cong \mathsf{S}(x, a)$.

2 Two Examples and Counting

So far, we've had two examples: $NISO$ and ISO. In this lecture we will dramatically enlarge the set of examples for which ZK proofs exists. To do this, we relax the requirement of statistical indistinguishability of the two distributions $VIEW_{\mathsf{V}'}^{\mathsf{PV}'}(x; x, a)$ and $\mathsf{S}(x, a)$, to computational indistinguishability.

Informally, this means that we no longer require that the simulator S (without interacting with anyone) generate exactly the same distribution (over all the possible transcripts) as the view of the verifier V (when interacting with the prover P). We just want them to "look" the same to any efficient algorithm (i.e. probabilistic polynomial time). If two distributions are computationally indistinguishable, then they look the same given a random poly-sized sample and probabilistic poly time. Note that if two distributions are statistically indistinguishable, then they look the same given a random poly-sized sample and an infinite amount of time, so statistical indistinguishability is strictly stronger than computational indistinguishability. Let's begin by defining computational indistinguishability more precisely.

2.1 Computational Indistinguishability

Probability Ensembles *A probability ensemble A is a family $A = \{A_k\}_{k \in \mathbb{N}}$ such that A_k is a probability distribution on some finite domain.*

Computational Indistinguishability *Let $\{A_k\}_{k \in \mathbb{N}}$ and $\{B_k\}_{k \in \mathbb{N}}$ be two probability ensembles. For any poly-sized family of tester circuits T, let $p_k^{A,T}$ be $Pr(\sigma \leftarrow A_k : T(1^k, \sigma) = 1)$ and $p_k^{B,T}$ be $Pr(\sigma \leftarrow B_k : T(1^k, \sigma) = 1)$. Then $\{A_k\}_{k \in \mathbb{N}}$ and $\{B_k\}_{k \in \mathbb{N}}$ are computationally indistinguishable under T iff for any positive polynomial $poly()$, and for all sufficiently large*

k's
$|p_k^{A,T} - p_k^{B,T}| < \frac{1}{poly(k)}$. *We write* $\{A_k\} \overset{T}{\approx} \{B_k\}$ *for computational indistinguishability under* T. *Iff for all poly-sized family of circuits* T, $\{A_k\} \overset{T}{\approx} \{B_k\}$, *then we say that* $\{A_k\}_{k\in\mathbb{N}}$ *and* $\{B_k\}_{k\in\mathbb{N}}$ *are computationally indistinguishable, and write* $\{A_k\}\approx\{B_k\}$.

Nonuniform vs Uniform Testers. Note that here we have stated computational indistinguishability under all poly-sized families of circuits instead of the usual all PPT Turing machines. This is in anticipation of what we would need later in the lecture. A poly-sized circuit family is just a collection of PPT Turing machines that have poly length descriptions, i.e. for each input length, you can have a different PPT Turing machine ("nonuniform" across the input lengths), whereas originally, we have a single machine for all the input lengths ("uniform").

In particular, a poly-sized family of circuits can have different polynomially long advice strings hardwired in it's circuit for different input lengths; the class of languages recognized by such families of circuits is termed $P/poly$. Later when we attempt to prove ZKness of a certain protocol, the verifier will play the role of a tester, and the advice it has is exactly a, the auxiliary input to it.

Is $P/poly$ more powerful than P? It's definitely not weaker. Even undecidable unary languages are in $P/poly$- since there is only one string of any given length, we simply hardwire an advice string indicating whether the corresponding unary string is in the language. So is $P/poly$ more powerful than NP? Probably not: if $NP \subseteq P/poly$, then the polynomial hierarchy collapses [KL80].

Discussion. Notice that two distributions can have completely different support (set of elements that are assigned positive probability), and yet be computationally indistinguishable. An example is the secure[1] encryptions of any pair of messages m_0 and m_1- if the ciphertexts produce weren't computationally indistinguishable, then the tester for it breaks the (secure) encryption.

3 A ZK proof system for an NP complete language

Now let's try to go beyond the two examples of ZK proof systems that we know. First we note that ZKness is a property of proof systems, so the underlying language must be efficiently provable, say an NP language. To get them all in one shot, let's try to give a ZK proof system for an NP complete language, graph 3-coloring. Then by first applying standard Cook reductions from the language at hand to 3-coloring and then executing the ZK proof system for 3-coloring, we can get ZK proof systems for all languages in NP.

The graph 3-coloring problem is, given a graph $G = (V, E)$, is there a way to color the vertices using only a total of 3 colorings so that no edge is connected to two vertices of the same color? More rigorously, does there exist a function $f : V \to \{1, 2, 3\}$ such that $f(i) \neq f(j)$ whenever $(i, j) \in E$?

Informally, if we have a map of the US, where the vertices are the cities and the edges are the roads connecting the cities, and P is trying to convince V that the graph so produced is 3-colorable, here is how the protocol works:

[1] "secure" as per the definition of Goldwasser and Micali [GM84]

1. P asks V to leave the room.
2. P 3-colors the graph using a randomly selected coloring scheme (out of a total of $3! = 6$), and covers the colorings of the vertices by putting paper plates on them. (We assume the prover knows a coloring of the graph, or finds one, and then permutes the colors randomly.)
3. V comes in, and picks a random edge, (i, j).
4. P uncovers the two plates covering the vertices i and j, thus revealing their colors.
5. V checks that they are of different colors. If not, V rejects.
6. They play this game $|E|^2$ times. Each time, P recolors the graph using a randomly selected coloring scheme, and V choses again some edge at random. If V gets to the end without rejecting, V accepts.

Now let's argue that this is a ZK proof system for 3-coloring.

1. **Efficiency:** Since what the verifier does is just to select random edges and check proper colorings, one basic iteration (steps 1 to 5) is efficient (for the verifier). Moreover, the number of edges is less than the input size, so we are simply repeating the game for a polynomial number of times.
2. **Completeness:** Clearly, if the graph was 3-colorable and P follows the protocol, then there will be no edges connecting two vertices of the same color, and thus V will be convinced with probability 1.
3. **Soundness:** If the graph is not 3-colorable, no matter what P' does, there is at least an edge that connects vertices of the same color. V will catch this with probability $\frac{1}{|E|}$ in each iteration of the game; over $|E|^2$ iterations, the probability of catching P is exponentially close to 1, i.e. the soundness error is exponentially small.
4. **ZKness:** Very informally, the simulator S simply puts paper plates on the vertices first (without coloring any of them). When V chooses some edge (i, j) at random (according to the random tape which S controls), S rewinds V, then colors i and j with different random colors (leaving the others as is), puts paper plates on all the verticies, and replays V, which again asks for the same edge. S reveals the edge, which passes the V's check.

3.1 Implementing the "Paper plates"

Now what is this "paper plate" thing in our argument? Notice that what we required was that, with the vertices covered by the plates, V has no clue of the coloring of the vertices. Also, with V present now in the room, P cannot recolor the vertices. The paper plates can be easily uncovered by P to show that the edge chosen by V is properly colored.

For simplicity, let's use a probabilistic Encryption scheme $(\mathsf{G}, \mathsf{E}, \mathsf{D})$ as our paper plate. Then the protocol would be:

1. P 3-colors the graph using a randomly selected coloring scheme. Say that we encrypt the colors R, W, B bit by bit, and the encoding is such that $R \to 00$ $W \to 01$, $B \to 10$, and 11 is unused. So if the first vertex was colored R, the second B,..., P sends over $\mathsf{E}(0)\mathsf{E}(0)\mathsf{E}(1)\mathsf{E}(0)$... (the concatenation of the various encryptions). In general, for vertex i, if b_i^1, b_i^2 is the encoding of it's color, then the i-th pair of ciphertexts sent to V will be $\mathsf{E}(b_i^1, R_i^1)\mathsf{E}(b_i^2, R_i^2)$ where R_i's are the random strings used in the encryption.

2. V receives $v = |V|$ pairs of strings, $(E_1^1, E_1^2, ..., E_v^1, E_v^2)$. Picks a random edge, (i, j), and sends over to P.

3. P rejects if garbage (i.e. an edge not in the graph) is received; else sends over $b_i^1, b_i^2, R_i^1, R_i^2, b_j^1, b_j^2, R_j^1, R_j^2$ thus revealing their colors.

4. V checks that they are of different colors: $(b_i^1, b_i^2) \neq (b_j^1, b_j^2)$.

   V checks that the revealing was proper:
   $\mathsf{E}(b_i^1, R_i^1) = E_i^1,$
   $\mathsf{E}(b_i^2, R_i^2) = E_i^2,$
   $\mathsf{E}(b_j^1, R_j^1) = E_j^1, and$
   $\mathsf{E}(b_j^2, R_j^2) = E_j^2.$

   V checks that the colors are not illegal: $(b_i^1, b_i^2) \neq (1,1)$ and $(b_j^1, b_j^2) \neq (1,1)$.

   If any check fails, V rejects.

5. They play this game $|E|^2$ times. Each time, P recolors the graph using a randomly selected coloring scheme, and V choses again some edge at random. If V gets to the end without rejecting, V accepts.

3.2 Proof of ZKness

Since the efficiency, completeness, and soundness all continues to hold, we only have to show that the proof system is Zero Knowledge. Again, we will actually prove something stronger – black-box Zero Knowledge – by reversing the quantifiers, namely that there exists $\mathsf{S}_{\mathsf{PPT}}$, for all $\mathsf{V}'_{\mathsf{PPT}}$, the simulated distribution is computationally indistinguishable from the view of $\mathsf{V}'_{\mathsf{PPT}}$:

$$\exists \mathsf{S}_{\mathsf{PPT}}, \forall \mathsf{V}'_{\mathsf{PPT}}, \forall x \in L, \forall a \in \{0,1\}^{poly(|x|)}, VIEW_{\mathsf{V}'}^{\mathsf{PV}'}(x; x, a) \approx \mathsf{S}^{\mathsf{V}'}(x, a).$$

S: On input (G, a), where G is a graph and a is the auxiliary input,

1. S generates at random a (PK, SK) pair using the generation algorithm of the encryption scheme.

2. S flips some amount of coins, and puts this random string, r, on V''s random tape; S passes the auxiliary input a, to V' as well.

3. S randomly selects an edge (i, j), and randomly chooses two distinct colors, one for i and the other for j. All the other vertices are, say, colored red.

4. S encrypts the colorings of the vertices properly (though most are red), exactly as is done by P, and sends the encryptions to V′ (together with the PK).

5. If V′ doesn't respond or sends garbage, S aborts, which is exactly what P would have done. Otherwise, S rewinds V′, and repeats the process until V′ happens to choose the same edge that it properly colored.

6. S outputs $(G, a, r, (E_1^1, E_1^2, ..., E_v^1, E_v^2), b_i^1, b_i^2, R_i^1, R_i^2, b_j^1, b_j^2, R_j^1, R_j^2)$. Note that we do not have to include the messages that V′ sent because these are completely determined by the inputs, the random string, and the messages that V′ has received.

Claims:

1. (Claim 0) If V′ sends garbage when interacting with S, whereas it sends a random edge when interacting with P, then this V′ can be used as a distinguisher for E(0) and E(1), which contradicts the security of the encryption scheme. (Basically, the only difference between what S sends and what P sends is that P encrypts an actual 3-coloring, while S does not. A simple hybridization between S's message and P's shows how to reduce this to breaking a bit encryption.)

2. (Claim 1) There is a probability of $\frac{1}{|E|}$ that V′ will choose the edge that S picked, so S will be able to proceed in the simulation. Again, if this is not true (or rather, if the probability is more than negligibly different from $\frac{1}{|E|}$), then V′ is able to distinguish what the simulator sends from what the prover sends, and thus break the encryption scheme.

   To present this hybrid argument with a little more detail, suppose that a 0-hybrid acts as the simulator and the $(|V|-2)$-hybrid acts as the real prover. The i-hybrid first picks a random edge and a random pair of good colors for the endpoints of that edge, then matches that up to the real 3-coloring to get a permutation of the colors. Then, the first i vertices in the graph (in some deterministic order) other than the two endpoints of the chosen edge are colored as they would be in the 3-coloring under the chosen permutation of the colors, while the remaining vertices in the graph would be colored red. Once all the colors are chosen, everything is encrypted, et cetera. Clearly, V′ chooses the given edge with probability $1/|E|$ given input from the $(|V|-2)$-hybrid since there is no information about that edge contained in the coloring at all. Thus, if the probability in the 0-hybrid case is significantly different there would be some pair of adjacent hybrids which can be distinguished.

   This should provide enough of a proof that the reader can fill in the rest independently. For claim 0, the only adjustment to this argument we need to make is that given the $(|V|-2)$-hybrid, the input is exactly the same as the real prover's input, so of course the probability is $1/2$.

   NOTE: It may seem like the simulator knows the coloring in this argument, but this is actually misleading. The way the logic works is this: we assume that a cheating verifier that falls into this case exists, and we assume there is a 3-colorable graph that the verifier works with. We prove this verifier can be used to break the underlying encryption, and our method for doing THIS knows the 3-coloring for that graph.

However, this is okay; our reduction is no longer trying to prove zero-knowledgeness, but is rather giving a contradiction.

3. (Claim 2) $VIEW_{\mathsf{V}'}^{\mathsf{PV}'}(x; x, a) \approx \mathsf{S}^{\mathsf{V}'}(x, a)$, conditional on the case that S was able to answer V's challenge.

   Again, the only difference between the output from the S when it succeeds and from the P is that the S encrypts something that is not a 3-coloring of the graph. As we have already shown, if any A can distinguish the two, A can be used to show that the underlying encryption scheme is not secure.

4 Concluding

We have seen that if secure encryption schemes exists, then all languages in NP have Zero Knowledge proof systems. Actually, bit commitment schemes suffices, and they exist iff one way functions exist, which is the lowest assumption one has to make in cryptography, since the existence of one way functions implies $P \neq NP$, which is still unresolved.

Next time, we are going to prove even more, namely that all languages in IP, the class of languages having interactive proofs, have Zero knowledge proof systems. So whatever you can prove, you can do so in Zero Knowledge.

We end by reflecting on what we've done. Do we really need to enlarge our hypothesis (i.e. change the requirement of statistical indistinguishability to computational indistinguishability) to get Zero Knowledge proof systems for all of NP? By a result of Fortnow [For89], we know that if $NP \subseteq PZK$ or $NP \subseteq SZK$[2], then the polynomial time hierarchy collapses, an event considered extremely unlikely. So, the step we took (going to computational zero-knowledge) is probably necessary to get all of NP.

References

[For89] L. Fortnow. The complexity of perfect zero-knowledge. In S. Micali, editor, Randomness and Computation, volume 5 of Advances in Computing Research, pp. 327-343, 1989.

[KL80] Richard M. Karp and Richard J. Lipton. Some connections between uniform and non-uniform complexity classes. Proc. 12th ACM Symp. on Theory of Computing, pp 302-309, 1980.

[GM84] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. Journal of Computer and System Sciences 28, pp. 270-299, 1984.

[2] PZK is the class of languages with perfect Zero knowledge proof systems, i.e. the simulator must generate exactly the same distribution, and SZK is the class of languages with statistical Zero Knowledge proof systems, i.e. the simulator must generate a statistically indistinguishable distribution from the verifier's view.