6.876/18.426: Advanced Cryptography Feb. 12, 2003

Lecture 3: Zero Knowledge Proofs Continued

Scribed by: Steve Weis

1 ZKP Definition Review

Recall from last lecture the definition of a Zero Knowledge Proof System (ZKPS). The goal of a ZKPS is to allow a prover P to prove to a probabilistic polynomial-time (PPT) verifier V that a given string x belongs to language L, without revealing any other information.

A ZKPS must have three properties: Completeness, Soundness, and Zero Knowledge. Completeness means that if $x \in L$, then a legitimate prover can always prove this to a legitimate verifier. Soundness means that verifiers cannot be fooled; if $x \notin L$, any prover (particularly malicious ones) cannot convince a verifier otherwise more than half the time. Finally, Zero Knowledge means that the verifier will only gain a single bit of knowledge: whether or not $x \in L$. We think of this as saying: "Anything I could learn by listening to P, I could just simulate myself". This PPT simulator (or subconscious) is denoted S below. Formally, these criteria are defined as follows:

1. **Completeness:** $\forall x \in L$ Prob$[(\mathsf{P},\mathsf{V})[\mathrm{x}] =$ "Yes"$] \geq 1 - neg(k)$.
2. **Soundness:** $\forall x \notin L\ \forall \mathsf{P}'$ Prob$[(\mathsf{P}',\mathsf{V})[\mathrm{x}] =$ "Yes"$] \leq \frac{1}{2}$
3. **Zero Knowledge:** $\forall \mathsf{V}'_{\mathsf{PPT}}\ \exists \mathsf{S}_{\mathsf{PPT}}\ \forall x \in L\ \ VIEW_{\mathsf{P},\mathsf{V}'}(x) = \mathsf{S}(x)$

This definition of zero knowledge is a bit weak. It is just saying that there is a specific simulator S for each possible verifier V'. To tighten this definition up, we would like to say that there is one simulator which could simulate *any* verifier/prover interaction. This motivates the following section.

2 Strong Zero Knowledge Definition

A stronger definition of the ZK criteria changes the order of quantifiers to require that there must exist a simulator capable of imitating any verifier. Since we can't build the definition of all verifiers into a simulator, we parameterize the verifier as an oracle. Oracles can be thought of as a remote procedure call call, since simulators have no insight in how the oracle actually works. A simulator S with oracle access to $\mathcal{O}$ is denoted $\mathsf{S}^{\mathcal{O}}$. The new definition is as follows:

Zero Knowledge: $\exists \mathsf{S}^{(\cdot)}_{\mathsf{PPT}}\ \forall \mathsf{V}'\ \forall x \in L\ \ VIEW_{\mathsf{P},\mathsf{V}'}(x) = \mathsf{S}^{\mathsf{V}'}(x)$

In the last lecture, we showed a ZKPS as defined in Section 1 for the Graph Isomorphism (ISO) problem. Recall that in ISO, you are given two graphs G and H and asked whether

they are isomorphic to each other. However, is the proof system we defined still a ZKPS under this new, stronger definition. The answer is yes, but how do we prove it?

Let's review the ISO ZKPS before proving that it holds under the stronger Zero Knowledge definition. Given input $(G_0, G_1) \in ISO$:

1. P randomly selects a graph C, isomorphic to both G_0 and G_1, and sends C to V.
2. V randomly selects either 0 or 1. (Suppose V chooses b). V sends b to P.
3. P finds a permutation π such that $C = \pi(G_b)$, and sends π to V.

Now consider how to simulate this with oracle access to V′. We will assume that our simulator S can write the contents of V′s random tape. A simulation works as follows:

1. S randomly selects 0 or 1. (Suppose b).
2. S randomly selects a random π_1 and sends $c_1 = \pi_1(G_b)$ to V′. Call V′s state α.
3. V′ randomly selects 0 or 1. (Suppose b'). V′ sends b' to S. V′ is now in state β
4. If $b = b'$ then S sends π_1 to V′. V′ is now in state γ. If not, S starts over.
5. S outputs the view V′ would have.

Through step 4, this looks like a legitimate iteration of the ZKPS. V′ will verify that $c_1 = \pi_1(G)$. Furthermore, the outputs from S are solely random choices, and will not yield any other information. However, if there were to be many repetitions of this protocol, chances are good the simulator would have to rewind at some point. If this happens we should be careful to rewind to a state where the verifier V′ has only seen good rounds. Since there is a 1/2 chance that S will choose the right graph, it will take an expected $2k$ trials before S can simulate k successful iterations.

There is one big problem: How do we "rewind" an oracle? S cannot access its innerworkings to reset its state to γ. Nor can S just continue a new iteration of the protocol, since this might skew the distribution of simulated view. S needs to get V′ in the exact same state before the protocol failed. The following section discusses how to do this.

3 Rewinding

A better description of rewinding might be "reset and fast forward". Since the simulator cannot just go into the verifier oracle and reset its state, it has to run the verifier from its initial state using identical input and random coins. There are three main approaches to doing this:

- **Code:** S has a copy of V′s code and can rewind simply by restarting V′ with the same inputs.
- **Black Box:** If S can't actually see the inner workings of V′, but has access to its random tape, and can restart the TM with the same random coins.
- **Clones:** If S doesn't even have access to V′s random tapes, it can create an unlimited number of clones of V′ and discard copies when they don't behave as desired.

4 Black Box ZKP

Oracle based simulations are referred referred to as Black-Box simulation, since it treats V as an immutable "black box". What if we give a universal S access to V's code? Does this have significantly more power? Or consider the first ZK definition: What if we are allowed to define S with respect to a given verifier V? These assumption settings may be described as follows:

1. **Black Box with Rewind:** $\exists \mathsf{S}_P PT\ \forall \mathsf{V}'_{\mathsf{PPT}}\ \ \mathsf{S}^{\mathsf{V}'}(x) = VIEW_{P,\mathsf{V}'}(x)$
   "A simulator can fool any verifier, even if he doesn't know how they work."

2. **Code Power:** $\exists \mathsf{S}_P PT\ \forall \mathsf{V}'_{\mathsf{PPT}}\ \ S(code(\mathsf{V}'), x) = VIEW_{P,\mathsf{V}'}(x)$
   "A simulator can fool any verifier, as long as he knows how they work."

3. **Full Power:** $\forall \mathsf{V}'_{\mathsf{PPT}}\ \exists S\ \ S(x) = VIEW_{P,\mathsf{V}'}(x)$
   "Verifier V_1 may be fooled by simulator S_1, and verifier V_2 may be fooled by simulator S_2, but S_1 and S_2 may work completely differently.

What do we gain from Full Power versus Code Power versus Black Box? Since a Black Box simulator has the "least" power, a proof system that is zero-knowledge under this definition has the strongest property. However, it is at least sensible to consider Code Power: perhaps oracle access will not be enough for some protocol, but given the verifier's code (which the *verifier* certainly has), we could simulate. Something that is Full Power zero-knowledge but not Code Power is somewhat hard to imagine. However, as we discussed last time, Full Power captures the right intuition for zero-knowledgeness, so we keep that as the core definition.

Where do we use the fact that V' is PPT except to specify the random tape? In this model, we don't. What happens if V' has some internal randomness, so that even clones behave differently? Then the Black-Box model fails to work. Each clone will behave differently in its initial state. What if V' is stronger, for example has exponential time? This is a difficult setting to formalize. We won't worry about this kind of detail here: this is where one should look at the original papers.

5 Another ZKP Example

So, we've shown one example of a ZKPS under our refined definition. Let's try to find another example, just to see how robust our definition is. This example is rather contrived: We will prove that an input is a natural number, that is $x \in \mathbb{N}$. First we need to review some tools from number theory that will help us.

5.1 Number Theoretic Review

We are going to take advantage of the difficulty of finding square roots of quadratic residues (that is, squares mod n). Given two odd, distinct primes, p_1 and p_2, let $n = p_1 p_2$. Let $x \in \mathbb{Z}_n$. Note that $x^2 \bmod n$ will have four square roots, denoted $x, -x, y$ and $-y$. Recall

from number theory the following:

$$GCD(x \pm y, n) = \begin{cases} p_1, \text{ or} \\ p_2 \end{cases}$$

This is by the following:

1. $x^2 \equiv y^2 \bmod n$
2. $x^2 - y^2 = (x+y)(x-y) \equiv 0 \bmod n$
3. $(x+y)(x-y) | kn$
4. $(x+y)(x-y) | p_1 p_2$

Without loss of generality, say that $p_1 | (x+y)$ and $p_2 | (x-y)$. Thus, if someone can find square roots, they can factor. As long as we think factoring is hard, we can assume that finding roots of quadratic residues is hard.

5.2 An Example ZKPS

Now let's design a ZKPS to prove memberships in the natural numbers. Define a language $L = \{n | n \in \mathbb{N}\}$. Let our proof system be as follows:

1. V selects a random $x \in \mathbb{Z}_n$ and sends P the value x^2.
2. P selects a random root $z \in \{x, -x, y, -y\}$ sends it to V.

This system is trivially complete and sound. Is it zero knowledge? Intuitively, no! V can find roots of squares mod n, which means that V could factor using P. But consider the $VIEW$: just the messages (x^2, z). A simulator could easily simulate this by selecting x itself and outputting (x^2, x). To an outside observer, S's view looks completely valid. This fits our definition of Zero Knowledge, but intuitively reveals a lot of knowledge. Which is wrong, our intuition or the definition? Let's hope the definition is. We need to refine it to accomodate this problem.

What's missing is V's choice of coins. What does our view look like if we inclode the coins, that is, what if $VIEW$ is a joint distribution over $(msgs, coins)$? Now legitimate views in the original ZKPS looks like: $((x^2, z), x)$. However the simulated view would always be $((x^2, x), x)$. Now someone will be able to see that the simulator always happens to output the same x value which V does, even though it should only occur half the time. Thus, the proof system as given, under a definition of $VIEW$ which includes random coins, is not Zero Knowledge. This definition is the one we will use from now on: $VIEW$ consists of all inputs and all random coins an algorithm gets; this includes not only messages but also any inputs given at the beginning of its execution.

6 Further refinement

So, this was one problem with our definition. What other properties would we like a ZKPS to have? For one, we would like a ZKPS to be reusable. It should still be a ZKPS after we run it once. With our current definition, this is not necessarily the case. Given a ZKPS (P, V), let $(\overline{\mathsf{P}},\overline{\mathsf{V}})$ run (P, V) twice. Now we have a problem, since $\overline{\mathsf{V}}'$ can take the results of the first iteration of V and give them to the second. A simulator S will not know what information may be passed between iterations.

(Note: our example for ISO is repeatable, and in a sense we have already proved that. However, the definition we had does not imply that *any* ZKPS is sequentially composable, which is a problem)

To address this issue, we will incorporate the concept of "advice" into our definition. Basically, both the simulator and verifier now have an additional parameter which signifies "prior knowledge", denoted a:

Zero Knowledge: $\forall \mathsf{V}'_{\mathsf{PPT}}\ \exists \mathsf{S}_{\mathsf{PPT}}\ \forall x \in L\ \forall a\ VIEW_{P,\mathsf{V}'(a)}(x) = \mathsf{S}(x, a)$.

Let us prove that zero-knowledge is composable with this definition. Suppose we have a verifier V' that engages in k copies of the protocol sequentially. We can split V' up into k parts: V'_1, $\mathsf{V}'_2, \ldots$, and imagine that instead of running V' over all k parts, we could just run V'_1, and then V'_2 with some state output from V'_1, and so on. This state output is just an advice string of some kind, so we can simulate the view of V' by running the simulator for V'_1 followed by the simulator for V'_2, and so on, each time giving the simulator the same advice string. Since each simulator is successful, this will produce an identically distributed view.

However, this idea of including the advice string in our definition is more important than just guaranteeing sequential composition. Imagine that there was a verifier V' for our ISO protocol, that if it is given half the permutation π that maps G_0 to G_1, can figure out the rest by talking to the prover. Clearly this is not zero-knowledge! Without the idea of the advice string, we only guarantee that verifiers that know nothing learn nothing. What we want is that no matter what verifier, no matter what information it knows, it gets no new knowledge.