Introduction to Cryptography 04/12/18

Lecture 20: Zero-Knowledge Proofs II

Instructor: Vipul Goyal *Scribe: Shaun Allison*

1 Review: Zero-Knowledge Proof of Graph Isomorphism

We briefly review the construction of the zero-knowledge proof of graph isomorphism from last lecture.

Suppose that the prover P wants to prove (in a zero-knowledge way) to verifier V that two graphs G_0 and G_1 are isomorphic. Suppose also that P has a witness $\pi : G_0 \cong G_1$ to the isomorphism.

P starts by choosing a random automorphism σ of G_0 and sending $\sigma(G_0)$ to V. The verifier responds with a random challenge $\text{Ch} \in \{0, 1\}$. If $\text{Ch} = 0$ then P responds with σ, otherwise P responds with $\sigma\pi^{-1}$. Finally, V checks that the function it just received is indeed an isomorphism between G_0 and the graph it received in the first step (i.e. $\sigma(G_0)$.)

To simulate transcripts between the prover and verifier, given the description V^* of a verifier, our simulator S will do the following:

- **(Step 1)**: Choose $b \in \{0, 1\}$ randomly and say that P sends $\sigma(G_b)$.
- **(Step 2)**: Get Ch from V^* and say that V sends Ch.
- **(Step 3)**: If $\text{Ch} = b$, say that P sends σ. Otherwise, restart.

We proved last time that this satisfies the Completeness and Zero-Knowledge properties, but for the Soundness property, we only showed that a cheating prover can only succeed against the honest verifier with probability at most one half. In order to get the probability down to a negligible amount, we will need to use a general technique to turn a zero-knowledge proof satisfying Soundness in this weaker sense, to a zero-knowledge proof satisfying the full Soundness proprety.

2 Soundness Amplification

Let's modify the scheme above for a given integer k like so:

- **(Step 1-1)**: P chooses a random automorphism σ_1 of G_0 and sends $\sigma_1(G_0)$.
- **(Step 1-2)**: V sends $\text{Ch}_1 \xleftarrow{\$} \{0, 1\}$.
- **(Step 1-3)**: If $\text{Ch}_1 = 0$ thet P sends σ_1, otherwise P sends $\sigma_1\pi^{-1}$.
- ...
- **(Step k-1)**: P chooses a random automorphism σ_k of G_0 and sends $\sigma_k(G_0)$.
- **(Step k-2)**: V sends $\text{Ch}_k \xleftarrow{\$} \{0, 1\}$.
- **(Step k-3)**: If $\text{Ch}_k = 0$ thet P sends σ_k, otherwise P sends $\sigma_k\pi^{-1}$.

In each step, V checks that P's responses all match up in the same way as before.

It turns out that this modification will satisfy the full soundness property, using the following lemma, which we will not prove (you can read on your own).

Lemma 1 ***(Soundness)*** *If G_0 is not isomorphic to G_1 then any cheating prover P^* fails with probability at least $1 - 2^{-k}$.*

We need to check that this modification still satisfies the Zero-knowledge property.

Lemma 2 *Sequential repitition is still zero-knowledge.*

Proof. Given a (possibly dishonest) PPT verifier V, we can produce a PPT algorithm S simulating transcripts in much the same way as before, using a new idea called "rewinding".

The simulator S will produce transcripts as follows:

1. **(Step 1-1)** Choose $b_1 \in \{0, 1\}$ at random. Choose σ_1 to be a random automorphism of G_{b_1}. Send $\sigma_1(G_{b_1})$.
2. **(Step 1-2)** Get Ch_1 from V^*.
3. **(Step 1-3)** If $Ch_1 = b_1$, send σ_1. Otherwise, restart.
4. **(Step k-1)** Choose $b_k \in \{0, 1\}$ at random. Choose σ_k to be a random automorphism of G_{b_k}. Send $\sigma_k(G_{b_k})$.
5. **(Step k-2)** Get Ch_k from V^*.
6. **(Step k-3)** If $Ch_k = b_k$, send σ_k. Otherwise, restart from **Step k-1**, restoring V's state to where it was before (this is called *rewinding*).

One can check in much the same way as last lecture that these transcripts are computationally indistinguishable from transcripts between the honest prover and verifier.

Rewinding is necessary here. If, instead of rewinding, the simulator restarted from the very beginning, it would become exponential, as the probability of making it to the end of all $3k$ steps will have probability 2^{-k}.

One can think of rewinding as making a copy of the state of V^* at every step, so if S ever fails, they can just restore the last copy of V^* and start again. S is allowed to have a complete description of V^*, and V^* is assumed to be a PPT algorithm, so this is possible. The intuition here is that V^* could have just generated the transcript themself, without having access to the witness, which means that the transcript gives no new knowledge.

This idea can in fact be applied to any zero-knowledge proof in much the same way:

Lemma 3 *Sequential repetition of any zero-knowledge proof yields a zero-knowledge proof.*

3 Foreshadowing: Graph 3-Coloring

In the next lecture, we will show how you can construct a zero-knowledge proof for Graph 3-Coloring, using a security assumption. Since Graph 3-Coloring is NP-complete, this will allow us to produce zero-knowledge proofs for all NP problems.

Graph 3-Coloring is the problem of deciding if the vertices of a given graph can be colored with only three colors, such that no two vertices of the same color are connected by an edge.

Before we present a zero-knowledge proof of Graph 3-Coloring, we need to introduce the idea of a commitment scheme. This idea will be important for the construction of zero-knowledge proofs, but they are also useful and interesting in their own right.

4 Commitment Schemes

Imagine that you can see the future, and you would like to prove this to your friend. So you write down tomorrow's lottery numbers on a piece of paper and lock it in a safe, and give the safe to your friend, away from your control. The next day, the lottery numbers are announced (as you predicted). You hand your friend the key to the safe, and they open it up and see that you indeed predicted today's lottery numbers. Since you were not able to alter the message since you wrote it yesterday, you have proven that you can see the future. This is an example of a use of commitment schemes.

A commitment scheme consists of a communication between a committer C and a receiver R.

- **Commitment Stage:** If the commiter wants to commit to a message m, C sends a string COM to R.
- **Decommitment Stage:** To allow R to open the message, C sends m and a string DECOM to R. R can either accept or reject.

A commitment scheme must satisfy two properties:

- **Hiding Property**: After the commitment stage, m is *semantically secure*, i.e. for all messages m_0 and m_1,
$$\{\text{COM} \leftarrow C(m_0)\} \approx_C \{\text{COM} \leftarrow C(m_1)\}$$
- **Binding Property**: Given COM, there exists exactly one m such that R outputs accept for (m, DECOM)

Some observations:

- The hiding property clearly requires C to be randomized.
- Consider this proposed commitment scheme: C chooses some string k at random and sends $\text{COM} = k \oplus m$. Then, when decommiting, C sends $\text{DECOM} = k$ and m.

  This scheme clearly satisfies hiding (it is a one-time-pad), however it does **not** satisfy binding. In fact, C doesn't commit to anything at all, as C can make the message look like anything they want to at the decommitment stage.

We will present two different constructions of commitment schemes: one using the Decisional Diffie-Hellman assumption, and one using any 1-1 one-way function.

4.1 Commitment Scheme from DDH

Let G be a prime-order group with generator g, and let q be the order of the group.

- **Commitment Stage**: C will choose a and b at random from $\mathbb{Z}_q$, and send

$$\text{COM} = (g, g^a, g^b, m \cdot g^{ab})$$

- **Decommitment Stage**: C will send

$$\text{DECOM} = (m, a, b).$$

  R will then check that $(g, g^a, g^b, m \cdot g^{ab})$ matches COM. If so, R will accept, otherwise reject.

Let's see that this satisfies the hiding and binding properties, using the DDH assumption.

- **Hiding Property**: This follows directly from the DDH assumption, but let's see this carefully, using properties of computational indestinguishability.

  The DDH assumption states that

$$\{(g, g^a, g^b, g^{ab} \mid (a, b) \xleftarrow{\$} \mathbb{Z}_q\} \approx_C \{(g, g^a, g^b, g^r \mid (a, b, r) \xleftarrow{\$} \mathbb{Z}_q\}.$$

  Using the fact that computational indistinguishability is preserved under PPT functions, for any message m,

$$\{(g, g^a, g^b, m \cdot g^{ab} \mid (a, b) \xleftarrow{\$} \mathbb{Z}_q\} \approx_C \{(g, g^a, g^b, m \cdot g^r \mid (a, b, r) \xleftarrow{\$} \mathbb{Z}_q\}.$$

  But

$$\{(g, g^a, g^b, m \cdot g^r \mid (a, b, r) \xleftarrow{\$} \mathbb{Z}_q\} = \{(g, g^a, g^b, g^r \mid (a, b, r) \xleftarrow{\$} \mathbb{Z}_q\}.$$

  So we have

$$\{(g, g^a, g^b, m \cdot g^{ab} \mid (a, b) \xleftarrow{\$} \mathbb{Z}_q\} \approx_C \{(g, g^a, g^b, g^r \mid (a, b, r) \xleftarrow{\$} \mathbb{Z}_q\}.$$

  Using the fact that computational indistinguishability is transitive, for any messages m_1 and m_2,

$$\{(g, g^a, g^b, m_1 \cdot g^{ab} \mid (a, b) \xleftarrow{\$} \mathbb{Z}_q\} \approx_C \{(g, g^a, g^b, m_2 \cdot g^{ab} \mid (a, b) \xleftarrow{\$} \mathbb{Z}_q\}.$$

  This is exactly what we need.

- **Binding Property**: When the commiter sends $\text{COM} = (g, g_1, g_2, g_3)$, we need to check that the message is fixed. But indeed, there are unique $a, b \in \mathbb{Z}_q$ such that $g_1 = g^a$, and $g_2 = g^b$. So a and b are fixed. So the message would have to be $g_3 \cdot (g^{ab})^{-1}$. So the message is fixed.

4.2 Blum Commitment

This construction, due to Manuel Blum, works for any 1-1 one-way function f with a hard-core bit. Recall that the existence of a one-way function implies the existence of a one-way function with a hard-core bit. This particular construction only allows the sending of a single bit, but it is clear that it can be modified to allow the commitment of longer messages.

- **Commitment Stage**: Let $m \in \{0,1\}$ be the message to commit. C will sample x uniformly at random from the domain of f. Then C will send

$$\text{COM} = (f(x), m \oplus h(x)),$$

  where h is a hard-core bit for f.

- **Decommitment Stage**: C will simply send

$$\text{DECOM} = (m, x).$$

  R will then compute $(f(x), m \oplus h(x))$ and check that it matches COM.

Let's see that this satisfies the definition of a commitment scheme.

- **Hiding Property**: This follows simply from the definition of hard-core predicate, which can be rewritten as

$$\{(f(x), h(x)) \mid x \xleftarrow{\$} \text{dom}(f)\} \approx_C \{(f(x), b) \mid x \xleftarrow{\$} \text{dom}(f), b \xleftarrow{\$} \{0,1\}\}.$$

- **Binding Property**: Follows immediately from the fact that f is 1-1.

What happens if we replace f with a one-way function that is not 1-1? The binding property will fail. But just because collisions exist, doesn't mean that they are easy to find. So we could design a commitment scheme using a collision-resistant hash function instead. But there is a subtle issue here – in order for the receiver R to trust that C could not have found collisions, R would need to have chosen the random seed themself. So we would need to modify our model to allow an extra round of communication during the commitment stage, where R sends C the random seed that they must use.

It is not known if there is a way to avoid this extra round of communication:

Open Problem 1 *Design a commitment scheme with one-round commitment and decommitment stages from the existence of any one-way function.*