Notes for Lecture 9

1 Last Time

Last time, we introduced zero knowledge proofs and showed how interactive zero knowledge proofs could be constructed from OWFs.

2 This Time

While zero knowledge proofs are interesting, an interactive protocol can be pretty limiting for some applications. Today, we will work towards zero knowledge proofs in a non-interactive protocol. We will introduce two new settings, the common random string CRM) model where the prover and verifier share some randomness, and the hidden bits model, which also deals with shared randomness, but restricts the random bits the verifier can see.

3 A Motivating example: CCA-secure PKE

Idea: Suppose $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is CPA-secure. We can define a new scheme where we run Gen twice.

$$(\mathsf{pk}_0, \mathsf{sk}_0) \leftarrow \mathsf{Gen}(), (\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathsf{Gen}()$$
$$((\mathsf{pk}_0, \mathsf{pk}_1), (\mathsf{sk}_0, \mathsf{sk}_1)) \leftarrow \mathsf{Gen}'()$$
$$\mathsf{Enc}'((\mathsf{pk}_0, \mathsf{pk}_1), m) = (\mathsf{Enc}(\mathsf{pk}_0, m), \mathsf{Enc}(\mathsf{pk}_1, m))$$

Decryption is defined analogously. We can try to prove CCA security using a hybrid argument. Let m_0, m_1 be a challenge, so an adversary needs to be able to distinguish between **Hybrid 0**, where the challenger encrypts m_0, and **Hybrid 2**, where the challenger encrypts m_1. This is differentiating between

$$\mathsf{Enc}'((\mathsf{pk}_0, \mathsf{pk}_1), m_0) = (\mathsf{Enc}(\mathsf{pk}_0, m_0), \mathsf{Enc}(\mathsf{pk}_1, m_0))$$

and

$$\mathsf{Enc}'((\mathsf{pk}_0, \mathsf{pk}_1), m_1) = (\mathsf{Enc}(\mathsf{pk}_0, m_1), \mathsf{Enc}(\mathsf{pk}_1, m_1))$$

We introduce **Hybrid 1** where the challenger returns

$$(\mathsf{Enc}(\mathsf{pk}_0, m_0), \mathsf{Enc}(\mathsf{pk}_1, m_1))$$

The idea here is to use an adversary A that can differentiate between **Hybrid 0** and **Hybrid 1** with nonnegligible probability to break CPA-security of $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. Unfortunately, this doesn't quite work, but we can make this work if we were able to include a zero knowledge proof that the ciphertext is well formed (proof that the same message is used for both parts). Unfortunately, for this to be of any use here, the proof must not involve interaction.

4 Settings for Non-interactive Zero Knowledge (NIZK)

Suppose we use the existing definition of zero knowledge, except restrict the communication to one message, π^* from the prover to the verifier. A prover P has x and a proof π and the verifier V just has x. Like our interactive zero knowledge proofs, the protocol must satisfy soundness and zero knowledge.

Soundness: If x is false, it's impossible for P to convince V that x is true.

Zero knowledge: There exists a simulator S that produces a view of the verifier that is computationally indistinguishable from the actual view.

$$\pi^* \leftarrow S(x) \approx_c \pi^* \leftarrow P(x, \pi)$$

These two properties can not both be fulfilled. If the prover just ran $S(x)$ and passed this to V, V would be fooled. In a sense, without interaction, our simulator is now too powerful. We get around this by introducing two new settings.

1. Common Random String (CRS) Model
   In this setting, the prover and verifier have access to shared randomness, $\mathsf{crs} \leftarrow \{0,1\}^n$.

2. Hidden Bits Model
   This is similar to the CRS model, but it's asymmetric. P has access to all the bits of crs, but this time sends $L \subseteq [n]$ to V, and V only has access to $\mathsf{crs}_L = (\mathsf{crs}_i)_{i \in L}$.

In both of these models,a simulator for the verifier will have it's own crs.

5 Construction of NIZK in Hidden Bits

We will construct NIZK for the Hamiltonian cycle problem. We can reduce other NP-complete problems to get NIKZs for specific problems.

Let $G = ([n], E)$, so all vertices are labeled by integers. P will have G along with a hamiltonian cycle c. V will just have the graph G.

We will assume that $\mathsf{crs} \in \{0,1\}^{n \times n}$ is a cycle matrix (all rows and all columns have exactly one 1. crs needs to be random, but we will address this later. The cycle represented by this matrix is σ

Let f be a random permutation $f : [n] \to [n]$ such that $f(c) = \sigma$

Protocol:

- P will send $\pi^* = f$ and $L = f(\bar{E})$, where $\bar{E}$ is all edges not in E.
- V will confirm that $L = f(\bar{E})$ and $\mathsf{crs}_L = 0^{|L|}$. If both are satisfied V will accept.

Soundness:
If G does not actually contain a Hamilitonian cycle c, we can't find an f such that $f(c) = \sigma$, as c must not be a cycle. Thus, for any f, $\mathsf{crs}_{f(\bar{E})}$ must contain a 1, and the V will reject.

Zero knowledge:
A simulator S chooses a random σ, so f will be a random permutation. From here V sees a random permutation f, $L = f(\bar{E})$, and crs bits that are all 0. If there was indeed a hamiltonian cycle, this is indistinguishable from an actual view of V.

We do have one problem that still needs to be solved. Our crs is still not actually random. This is dealt with by choosing a $\mathsf{crs} \in \{0,1\}^{n^4}$ where each bit of the matrix is 1 with probability $\frac{1}{n^3}$. We can then consider crs' to be the subset of rows and columns containing exactly one 1. We claim (without proof) that the resultant matrix is a cycle matrix with probability $> \frac{1}{n^3}$. [1]. This results in very poor soundness, but we can boost our soundness via parallel repetition ($n^3\lambda$ times should do). As usual, we should be careful that repetition does not break zero knowledge. In this case, it's

[1] This can be proved using Chebychev's inequality to lower bound the probability of having exactly n ones, and then handle the probability that each row and column has more than one 1. See http://www.cs.umd.edu/ jkatz/gradcrypto2/NOTES/lecture13.pdf, for a similar result

okay, because we are in a non-interactive setting and can collect all our messages and send at once.

We do still need to resolve how we generate a 1 with probability $\frac{1}{n^3}$ for our crs. This can be done with $\log(n^3)$ bits for each index in our crs matrix. All of these bits will be 1 with probability $\frac{1}{2^{\log(n^3)}} = \frac{1}{n^3}$. All in all, our $\mathsf{crs} \leftarrow \{0,1\}^{n^7 \lambda \log(n^3)}$.

6 Conversion of hidden bits NIKZ to CRS NIKZ

We will now show how to tranlate a hidden bits protocol to CRS. Let P_{HB}, V_{HB} be the hidden bits prover and verifier. Suppose we have a $\mathsf{OWP}, p : \{0,1\}^\lambda \to \{0,1\}^\lambda$, with a hardcore bit h. Let $\mathsf{crs} \leftarrow (\{0,1\}^\lambda)^n$

Prover $P(x, \pi)$:

1. Compute $\alpha_i \leftarrow p^{-1}(\mathsf{crs}_i)$.
2. Let $\mathsf{crs}_{HB} = h(\alpha_i)$.
3. Run $\pi^*_{HB} \leftarrow P_{HB}(x, \pi, \mathsf{crs}_{HB})$
4. Send verifier $(\pi^*_{HB}, L, \{\alpha_i\}_{i \in L}, \{\mathsf{crs}_{HBi}\}_{i \in L})$

Verifier $V(x, (\pi^*_{HB}, L, \{\alpha_i\}_{i \in L}, \{\mathsf{crs}_{HBi}\}_{i \in L}))$:

1. Check that $p(\alpha_i) = \mathsf{crs}_i$.
2. Check that $\forall i \in L, h(\alpha_i) = (\mathsf{crs}_{HB})_i$.
3. Run $V_{HB}((\mathsf{crs}_{HB})_L, x, \pi^*_{HB})$

The intuition of this protocol is that all hidden bits are now hardcore bits. Soundness follows from the soundness of the original protocol and the fact that the α_i and crs_{HB} are verified against the crs. Zero knowledge follows by thinking of α_i as random, and the idea that applying a permutation to random bits produces random bits.

Simulator S:

1. Run $(\pi^*, L, (\mathsf{crs}_{HB})_L) \leftarrow S_{HB}(x)$
2. For $i \in L$ choose α_i such that $h(\alpha_i) = (\mathsf{crs}_{HB})_i$

3. for $i \notin L$ chose α_i at random
4. $\mathsf{crs}_i = p(\alpha_i)$
5. Output $(\mathsf{crs}, \pi^*, \{\alpha_i\}_{i \in L}, (\mathsf{crs}_{HB})_L)$

7 Making the prover efficient

While the previous construction works, it wouldn't work in practice because the prover is inefficient. This is because the prover is required to invert p. This can be solved by using a trapdoor permutation.

Definition 1 *A trapdoor permutation is a triple* $(\mathsf{Gen}, p, p^{-1})$ *where* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$*, where* $y \leftarrow p(\mathsf{pk}, x)$ *is a permutation and* $x \leftarrow p^{-1}(\mathsf{sk}, y)$*. For security, we need* $p(\mathsf{pk}, \cdot)$ *to be a* OWP *for a random* pk*.*

Replacing all uses of the permutation p with trapdoor permutations (and adding pk to the proof) gets us an efficient prover.

Prover $P(x, \pi)$:

1. $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen}(1^\lambda)$
2. Compute $\alpha_i \leftarrow p^{-1}(\mathsf{sk}, \mathsf{crs}_i)$.
3. Let $\mathsf{crs}_{HB} = h(\alpha_i)$.
4. Run $\pi^*_{HB} \leftarrow P_{HB}(x, \pi, \mathsf{crs}_{HB})$
5. Send verifier $(\mathsf{pk}, \pi^*_{HB}, L, \{\alpha_i\}_{i \in L}, \{\mathsf{crs}_{HBi}\}_{i \in L})$

Verifier $V(x, (\pi^*_{HB}, L, \{\alpha_i\}_{i \in L}, \{\mathsf{crs}_{HBi}\}_{i \in L}))$:

1. Check that $p(\mathsf{pk}, \alpha_i) = \mathsf{crs}_i$.
2. Check that $\forall i \in L, h(\alpha_i) = (\mathsf{crs}_{HB})_i$.
3. Run $V_{HB}((\mathsf{crs}_{HB})_L, x, \pi^*_{HB})$

8 Construction of Trapdoor Permutation

There is only one known construction of a trapdoor permutation.

- Gen chooses large primes p, q, where $N = pq$.
- Choose a random e such that $\gcd(e, (p-1)(q-1)) = 1$
- $d = e^{-1} \mod (q-1)(p-1)$
- Let $\mathsf{pk} = (N, e), \mathsf{sk} = (N, d)$
- $p(\mathsf{pk}, x) = x^e \mod N$
- $p^{-1}(\mathsf{sk}, y) = y^d \mod N$

This defines a trapdoor permutation over $[N]$.