COS 533: Advanced Cryptography | Princeton University
Lecture 11 (March 11, 2021)
Lecturer: Mark Zhandry | Scribe: Qinhan Tan

# Notes for Lecture 11

## 1 Introduction

Last time, we introduced Non-interactive Zero-knowledge (NIZK). We talked about the CRS Model and Hidden Bits Model, then showed how to translate the Hidden Bits Model into a CRS Model.

Today, we are going to construct NIZK in Hidden Bits Model for all languages in NP. We will first contruct a NIZK for Hamoltonian cycle. Then, using NP reductions, this gives a NIZK for all NP problems.

After NIZK, we will talk about Identity-based Encryption.

## 2 NZIK for Hamiltonian Cycle

Given a graph $G = (V, E)$, it has a Hamiltonian cycle if there is a cycle that visits all nodes without repeats. The Hamiltonian cycle problem is NP-complete.

We are going to construct a protocol in the Hidden Bits Model where a prover ($P$) can convince a verifier ($V$) that a graph $G$ has a Hamiltonian cycle. Assume that $P$ knows $G$ and a Hamiltonian cycle, while $V$ $C$[1], while $V$ only knows $G$. Everything from here will be efficient.

**Protocol:**

Recall that in the Hidden Bits Model, $P$ knows a $crs$ which should be a uniformly random bit string, while $V$ only knows a subset of $crs$ specified by a set of locations $L$ determined by $P$. We will cheat a little bit for now and return to this later. Assume that $crs \in \{0,1\}^{n\times n}$ is a random cycle matrix. It's like an adjacency matrix of a graph that is just a cycle. Let $\sigma$ denote the cycle represented by $crs$.

$P$ is going to choose a random permutation $f$ on labels of end vertices of edges, $f : [n] \to [n]$ $s.t.$ $f(c) = \sigma$. This means $f$ maps Hamiltonian cycle $c$ into a cycle that is a part of $crs$. The proof $\pi$ that $P$ is going to send to $V$ is $f$, and $L = f(\bar{E}))$ ($\bar{E}$ is the set of all edges not in $E$). Basically, $P$ takes the graph $G$, permutes it under $f$,

[1] $P$ can be inefficient and do brute-force search for the Hamiltonian cycle, or it can be given a Hamiltonian cycle

and then the set of positions of $crs$ it will reveal is exactly the set of edges that are the images of edges under $f$ that aren't in the graph. Note that for a honest $P$, the only 1's in the CRS are the images of the Hamiltonian under $f$ by the requirement that $f(c) = \sigma$. All edges outside of the graph are 0 so the $crs_L$ that's revealed to $V$ is all 0's. $crs_L$ would be a length $|L|$ string of all 0's.

After receiving $\pi, L, crs_L$, $V$ will output 1 iff. $L = f(\bar{E})$ and $crs_L = 0^{|L|}$.

**Soundness:**

Suppose $G$ does not contain a Hamiltonian cycle, then there does not exists a $f$ such that $f(E)$ covers $c$. In other words, $\forall f, \exists e \in \sigma \; s.t. \; e \notin f(E)$, which is equivalent to $e \in f(\bar{E})$. Then $crs_{f(\bar{E})}$ will contain at least one 1. Thus, $V$ will reject.

**Zero-knowledge:**

Recall that we first choose a random cycle $\sigma$, and choose a random permutation that maps the particular Hamiltonian cycle to $\sigma$. This is equivalent to choosing a truly random permutation $f$ and let $\sigma = f(c)$. Thus, what $V$ sees are a random permutation $f$, $f(\bar{E})$ and $0^{|E|}$. We can construct the following simulator:

$S(G)$: choose a random permutation $f$ and output $(f, f(\bar{E}), 0^{|E|})$.

The only left issue is that $crs$ is not a truly random string, but has some structure. We have to modify things to allow us to mimic the structure with a truly random string. The solution is to choose $crs \leftarrow \{0,1\}^{n^2 \times n^2}$ where each bit is i.i.d. such that each bit is 1 with probability $\frac{1}{n^3}$ (not uniformly random). Then, let's define $crs'$ as the subset of rows and columns containing a 1. We can make a claim (we won't prove it) that $crs'$ is a cycle matrix with probability at least $\frac{1}{n^3}$ [2]. Now, we will make a few modifications to the above protocol.

(1) If $crs'$ exists, then $P$ will use $crs'$ in the protocol and reveal all bits outside of $crs'$. $V$ will verify that all the bits outside of $crs'$ are 0. If there exists the Hamiltonian cycle, it must be within $crs'$ and then the soundness of the protocol works.

(2) If $crs'$ does not exist, then the prover will set $\pi = \{\}$ and $L$ to be everything (reveal all bits of $crs$). $V$ checks that $crs$ does not consist of a cycle of length $n$.

Zero-knowledge is straightforward from above. In case (1), $V$ sees a random permutation $f$ and a bunch of 0's. In case (2), $V$ sees a random malformed crs. Both cases can be simulated. Soundness is a big issue. If $crs$ is not well formed, $P$ wo;; trivially convince the verifier. This means we can only catch a cheating $P$ with probability

[2] The intuition is that we have $n^4$ different bits where each bit has $\frac{1}{n^3}$ probability to be 1, so in expectation there are $n$ 1's. Then with good probability all 1's will be in distinct rows and columns. With a sort of careful analysis, we can show that the probability that $crs'$ being a cycle is about $\frac{1}{n^3}$

$\frac{1}{n^3}$, when $crs$ is well formed. The solution is to repeat $\lambda n^3$ times to reduce the cheating probability to $2^{-O(\lambda)}$. For NIZKs, parallel and sequential repetition are identical, which is different from interactive zero knowledge[3].

The last piece is that $crs$ is still not uniformly random, so we need to generate these i.i.d. bits that are 1 with probability $\frac{1}{n^3}$. For each of these bits, we will generate $\log(n^3)$ uniformly random bits, set the bit to be the logical AND of all of them. To reveal a logical bit, $P$ just reveals the corresponding $\log(n^3)$ real bits. Thus, the actual $crs \in \{0,1\}^{n^3\lambda\times n^2\times n^2\times\log(n^3)}$. This completes the proof.

# 3 Identity-based Encryption (IBE)

With public key encryption, you need to tell everyone your public key. This causes difficulty in recording long public keys. In IBE, the public key is just a bit-string that represents one's identity which might be email address or phone number. There's no restriction on the length of the identity because you can use a collision resistant hash function to hash your identity.

To be able to decrypt using a secret key other than the identity, IBE needs a trusted authority who will run a setup algorithm to give all of the users their specific secret key. The setup algorithm will generate a master public key $mpk$ and a master secret key $msk$.

**Syntax**:

$$\begin{aligned}(mpk, msk) &\leftarrow \mathsf{Gen}(1^\lambda)\\ c &\leftarrow \mathsf{Enc}(mpk, id, m)\\ sk_{id} &\leftarrow \mathsf{Extract}(msk, id)\\ m &\leftarrow \mathsf{Dec}(sk_{id}, c)\end{aligned}$$

.

**Correctness:**

$$\forall (mpk, msk) \leftarrow \mathsf{Gen}(1^\lambda), id, m, sk_{id} \leftarrow \mathsf{Extract}(msk, id)$$
$$\Pr[\mathsf{Dec}(sk_{id}, \mathsf{Enc}(mpk, id, m)) = m] = 1$$

**Security:**

The security goal is that no users but $id$ decrypt messages to $id$. We also want collusion resistance in case some users collude, where an adversary who are actually a group of users may use multiple $sk_{id}$ to hack another user. The security game is described as follows:

(1) The chanllenger $Ch$ runs $(msk, mpk) \leftarrow Gen(1^\lambda)$ and sends $mpk$ to adversary

[3]We have a statement that for zero knowledge you can't necessarily do parallel repetition and preserve their own knowledge. You can only do sequential repetition and preserve their knowledge.

$A$.

(2) $A$ sends an $id$ to $Ch$.

(3) $Ch$ responses with $sk_{id} \leftarrow \mathsf{Extract}(msk, id)$.

(4) $A$ may repeat the $sk$ query in (2) for many times get responses as in (3). Then at some point, $A$ will send a challenge query $(id^* \notin \{id \text{ queried so far}\}, m_0, m_1)$.

(5) $Ch$ responses with $c^* \leftarrow \mathsf{Enc}(mpk, id^*, m_b)$.

(6) $A$ will try to guess $b$ and output $b'$. During this period, $A$ is still allowed to do (2) and gets a $sk_{id}$ as long as $id \neq id^*$.

Because $A$ can encrypt messages by themselves, $\mathsf{Enc}$ needs to be randomized.

**Trivial Construction:**

When there are only a polynomial number of $id$s, namely $id \in 0, \ldots, t = poly(\lambda)$, there is a trivial solution:

$\mathsf{Gen}(1^\lambda)$: Generate $t$ $(pk_i, sk_i)$ for a PKE scheme. $mpk = \{pk_i\}, msk = \{sk_i\}$

$\mathsf{Extract}(msk, id) = sk_{id}$

$\mathsf{Enc}(mpk, id, m) = \mathsf{Enc}_{PKE}(pk_{id}, m)$

$\mathsf{Dec}(sk_{id}, c) = \mathsf{Dec}_{PKE}(sk_{id}, c)$

The security simply follows from the security of the PKE scheme. This trivial example shows that what makes IBE challenging is IBE needs to compress exponentially many public keys into a single $mpk$.

The following is a sketch of an IBE for large identities, but without collusion resistance:

$mpk$ is a $2 \times n$ grid of public keys of a PKE: $pk_{10}, pk_{11}, pk_{20}, pk_{21} \ldots pk_{n0}, pk_{n1}$, and $msk$ is the grid of corresponding secret keys.

For $\mathsf{Enc}(mpk, id \in \{0,1\}^n, m)$, we choose random $m_i s.t. \oplus_{i=1}^n m_i = m$. Let $c_i = \mathsf{Enc}_{PKE}(pk_{i,id_i}, m_i)$.

The idea is that in encryption, the identity $id$ is used to select public keys to do encryptions. And the secret key holder for identity $id$ will have the corresponding secret keys for the selected public keys and will be able to recover all $m_i$s, while any other identity will not be able to recover all $m_i$s.

# 4 Next

Using algebraic constructions to achieve collusion resistance for large identities.