CS 600.442 – Modern Cryptography 11/02/2016

Lecture 1: Non-Interactive Zero Knowledge (II)

Instructor: Abhishek Jain *Scribe: Alex Badiceanu*

Recall that our goal is to construct NIZKs for NP in the CRS model. Last time we defined NIZKs in the hidden bit (HB) model and we showed a transformation from NIZKs in HB model to NIZKs in the CRS model. Today, we will construct NIZKs for NP in the HB model. In particular, we will construct NIZKs for the Graph Hamiltonian problem in the HB model.

We start by defining Hamiltonian graphs and some related notions.

1 Hamiltonian Graphs

A Hamiltonian graph is a graph that consists a Hamiltonian cycle. In other words, there exists a cycle formed by edges in the graph that visits each vertex exactly once. More formally:

Definition 1 (Hamiltonian Graph) *Let $G = (V, E)$ be a graph with $|V| = n$. We say that G is a Hamiltonian graph if it has a Hamiltonian cycle, i.e. there are $v_1, ..., v_n \in V$ such that for all $i \in [n]$:*

$$(v_i, v_{(i+1) \mod n}) \in E$$

Fact: Deciding whether a graph is Hamiltonian in **NP**-Complete. Let L_H be the language of Hamiltonian graphs $G = (V, E)$ s.t. $|V| = n$

Any graph can be represented as an adjacency matrix. The number of rows and columns of this matrix is the same as the number of vertices in the graph. A value of 1 at a given position represents the presence of an edge between the vertices corresponding to the row and column. More specifically:

Definition 2 (Adjacency Matrix) *A graph $G = (V, E)$ with $|V| = n$, can be represented as an $n \times n$ adjacency matrix M_G of boolean values such that:*

$$M[i,j] = \begin{cases} 1 & \textit{if } (i,j) \in E \\ 0 & \textit{otherwise} \end{cases}$$

Definition 3 (Cycle Matrix) *A cycle matrix is a boolean matrix that corresponds to a graph that contains a Hamiltonian cycle and no other edges.*

Definition 4 (Permutation Matrix) *A permutation matrix is a boolean matrix such that each row and each column has exactly one entry equal to 1.*

Note: Every cycle matrix is a permutation matrix, but the converse is not true. For every n, there are $n!$ permutation matrices, but only $(n-1)!$ cycle matrices.

Note that a consequence of the above is that if we pick a permutation matrix at random, it is also a cycle matrix with probability $\frac{1}{n}$.

2 NIZKs for L_H in Hidden-Bit Model

We want to come up with NIZKs for the Hamiltonian graph problem in the HB model. We are going to do this in two steps:

Step I: NIZK (K_1, P_1, V_1) for L_H in hidden-bit model where K produces (hidden) strings r with a specific distribution: each r represents an $n \times n$ cycle matrix.

This first step is a simplified case. In the HB model we had a truly random string. The prover gets to see this random string, but the verifier only gets to see some part of this random string that is decided by the prover. What we are doing in the first step is considering a simplified model where we will allow the string to have a very particular distribution. It will not be truly random, but biased. In particular, we will consider NIZKs for the Hamiltonian graph problem where the algorithm will produce strings that will represent a $n \times n$ matrix.

Step II: Modify the above construction to obtain $K_2, P_2, V_2)$ where the (hidden) string r is uniformly random

Once we have the construction from the previous step, we will show how to extend this construction so that we can move to the real world where the string is supposed to be uniform. At high level, we will use a very large random string and the come up with some deterministic algorithm which will give us a way to convert such a long random string into a small random string which will have the desired distribution with very high probability.

2.1 Step I

Construction of $K_1, P_1, V_1)$ for L_H: We describe the algorithms below.

$K_1(1^n)$: Output $r \leftarrow \{0,1\}^{n^2}$ s.t. it represents an $n \times n$ cycle matrix M_C
In other words, K takes the security parameter and outputs a string r with a very particular distribution such that r represents a cycle matrix. This represents the input given to the prover.

$P_1(r, x, w)$: Execute the following steps:

- Parse $x = G = (V, E)$ s.t. $|V| = n$, and $w = H$ where $H = (v_1, ..., v_n)$ is a Hamiltonian cycle in G. To be more specific, instance X corresponds to a graph supposedly Hamiltonian, and w is some witness represented by a Hamiltonian cycle in the graph.
- Choose a permutation $\varphi : V \to \{1, ..., n\}$ that maps H to the cycle in M_C, i.e., for every $i \in [n]$:

$$M_C[\varphi(v_i), \varphi(v_{(i+1) \mod n})] = 1$$

- Define $I = \{\varphi(u), \varphi(v) | M_G[u, v] = 0\}$ to be the set of non-edges in G
- Output (I, φ)

To summarize, the prover first picked a permutation that maps the witness to the cycle in the random string r. It wants to be given some evidence that it actually has a cycle. It is going to show that all non-edges in $\varphi(G)$ are mapped to a 0 value. The verifier can accomplish this task. The idea is that if G does not actually have a cycle, then no matter what mapping we come up with, at least one non-edge in G will get mapped to an edge in the cycle graph and then the verifier will catch it. (M_G is the adjacency matrix of G).

$V_1(I, r_I, \varphi)$: Execute the following steps:

- Parse $r_I = \{M_C[u,v]\}_{(u,v)\in I}$
- Check that for every $(u,v) \in I, M_C[u,v] = 0$
- Check that for every $(u,v) \in I, M_G(\varphi^{-1}(u), \varphi^{-1}(v)) = 0$
- If both the checks succeed, then output 1 and 0 otherwise

Completeness: An honest prover P can always find a correct mapping φ that maps H to the cycle in M_C.

Soundness: If $G = (V, E)$ is not a Hamiltonian graph, then for any mapping $\varphi \to \{1, ..., n\}, \varphi(G)$ will not cover all the edges in M_C. There must exist at least one non-zero entry in M_C that is revealed as a non-edge of G.

Zero Knowledge: Simulator S performs the following steps:

- Sample a random permutation $\varphi : V \to \{1, ..., n\}$
- Compute $I = \{\varphi(u), \varphi(v) | M_G[u,v] = 0\}$
- For every $(a,b) \in I$, set $M_C[a,b] = 0$
- Output $(I, \{M_C[a,b]\}_{(a,b)\in I}, \varphi)$

It is easy to verify that the above output distribution is identical to the real experiment.

Note that here, the simulator can choose the set r_I, so it controls the string r seen by the verifier. Therefore, this simulator is non-adaptive. When we transform such a NIZK in HB model to a NIZK in the CRS model, this non-adaptivity property carries over which makes the CRS non-reusable.

2.2 Step II

We start by describing the strategy in this step:

- Define a deterministic procedure Q that takes as input a (polynomially long) random string r and outputs a biased string s that corresponds to a cycle matrix with inverse polynomial probability $\frac{1}{l(n)}$. We want to come up with a way to amplify this probability and make it closer to 1.
- If we feed $Qn \cdot l(n)$ random inputs, then with high probability, at least one of the outputs will correspond to a cycle matrix

- In the NIZK construction, the (hidden) random string will be $r = r_1, \ldots, r_{n \cdot l(n)}$
- For every i, the prover will try to compute a proof using $s_i = Q(r_i)$. In other words, it will take a chunk from the random string r_I, apply the deterministic procedure Q on it to obtain some string s_i, and now with this much probability s_i will be a cycle matrix
- At least one s_i will contain a cycle matrix, so we can use the NIZK proof system from Step 1

We now explain the deterministic procedure Q.

Procedure $Q(r)$:

- Parse $r = r_1, ..., r_{n^4}$ s.t. $\forall i, |r_i| = \lceil 3 \log n \rceil$
- Compute $s = s_1, ..., s_{n^4}$, where:

$$s_i = \begin{cases} 1 & \text{if } r_i = 111 \cdots 1 \\ 0 & \text{otherwise} \end{cases}$$

- Define an $n^2 \times n^2$ boolean matrix M consisting of entries from s
- If M contains an $n \times n$ sub-matrix M_C s.t. M_C is a cycle matrix, then output (M, M_C), else output $(M, \perp)$.

Analysis of Q: Let GOOD be the set of outputs of $Q(\cdot)$ that contain a cycle matrix and BAD be the complementary set.

Lemma 1 *For a random input r, $PR[Q(r) \in GOOD] \geq \frac{1}{3n^3}$*

Let M be an $n^2 \times n^2$ matrix computed by Q on a random input r. We will prove the above lemma via a sequence of claims:

Claim 1: M contains exactly n 1's with probability at least $\frac{1}{3n}$
Claim 2: M contains a permutation sub-matrix with probability at least $\frac{1}{3n^2}$
Claim 3: M contains a cycle sub-matrix with probability at least $\frac{1}{3n^3}$

Proof of Claim 1: Let X be the random variable denoting the number of 1's in M.

- X follows the binomial distribution with $N = n^4, p = \frac{1}{n^3}$
- $E(X) = N \cdot p = n$
- $Var(X) = Np(1-p) < n$
- Recall Chebyshev's Inequality: $Pr[|X - E(X)| > k] \leq \frac{Var(X)}{k^2}$
  Setting $k = n$, we have:

$$Pr[|X - n| > n] \leq \frac{1}{n}$$

- Observe:

$$\sum_{i=1}^{2n} Pr[X=i] = 1 - Pr[|X-n| > n] > 1 - \frac{1}{n}$$

- $Pr[X=i]$ is maximum at $i=n$
- Observe:

$$Pr[X=n] \geq \frac{\sum_{i=1}^{2n} Pr[X=i]}{2n}$$

$$\geq \tfrac{1}{3n}$$

Proof of Claim 2: We want to bound the probability that each of the n '1' entries in M is in a different row and column.

- After k '1' entries have been added to M,

$$Pr[\text{new '1' entry is in different row and column}] = \left(1 - \tfrac{k}{n^2}\right)^2$$

- Multiplying all:

$$Pr[\text{no collision}] \geq \left(1 - \tfrac{1}{n^2}\right)^2 \cdots \left(1 - \tfrac{n-1}{n^2}\right)^2$$

$$\geq \tfrac{1}{n}$$

- Combining the above with Claim 1:

$$Pr[\text{M contains a permutation } n \times n \text{ submatrix}] \geq \tfrac{1}{3n^2}$$

Proof of Claim 3 We want to bound the probability that M contains an $n \times n$ cycle sub-matrix

- Observe:

$$Pr[n \times n \text{ permutation matrix is a cycle matrix }] = \tfrac{1}{n}$$

- Combining the above with Claim 2,

$$Pr[M \text{ contains a cycle } n \times n \text{ submatrix } \geq \tfrac{1}{3n^3}$$

Construction of (K_2, P_2, V_2) for L_H: We now describe the algorithms

$K_2(1^n)$: Output $r \leftarrow \{0,1\}^L$ where $L = \lceil 3 \log n \rceil \cdot n^8$

$P_2(r, x, w)$: Parse $r = r_1, ..., r_{n^4}$ s.t. for every $i \in [n^4], |r_i| = \lceil 3 \log n \rceil \cdot n^4$.
For every $i \in [n^4]$:

- If $Q(r_i) = (M^i, \perp)$,set $I_i = [|r_i|]$(i.e., reveal the entire r_i), and $\pi_i = \emptyset$
- Else, let $(M^i, M_C^i) \leftarrow Q(r_i)$. Compute$(I_i', \varphi_i) \leftarrow P_1(M_C^i, x, w)$. Set $I_i = I_i' \cup J_i$ where J_i is the set of indices s.t. r_i restricted to J_i yields the residual M^i after removing M_C^i, and $\pi_i = \varphi_i$

Output $(I = \{I_i\}, \pi = \{\pi_i\})$

$V_2(I, r_I, \pi)$: Parse $I = I_1, ..., I_{n^4}$, and $\pi = \pi_1, ..., \pi_{n^4}$. For every $i \in [n^4]$:

- If I_i is the complete set, then check that $Q(s_i) = (\cdot, \perp)$
- Otherwise, parse $I_i = I_i' \cup J_i$. Parse $s_i = s_i^1, s_i^2$ and check that s_i^2 is the all 0 string. Also, check that $V_1(I_i', s_i^1, \pi_i) = 1$.

If all the checks succeed, then output 1 and 0 otherwise.

Completeness: It follows from completeness of the construction in Step I.

Soundness: For random $r = r_1, ..., r_{n^4}, Q(r_i) \in$ GOOD for at least one r_i with high probability. Soundness then follows from the soundness of the construction in Step I.

Zero-Knowledge: For i s.t. $Q(R_i) \in$ GOOD, V does not learn any information from the zero-knowledge property of the construction in Step 1. For i s.t. $Q(r_i) \in$ BAD, V does not see anything besides r_i.