Introduction to Cryptography 4/17/18

Lecture 21: Zero-Knowledge Proofs III

Instructor: Vipul Goyal *Scribe: Colin Kelly*

1 3-colorable Graphs

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.

Definition 1 *A graph G is 3-colorable 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.*

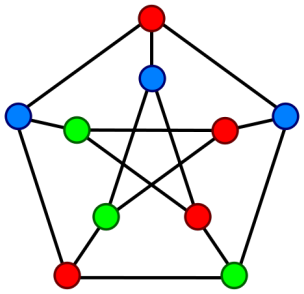

Figure 1: A 3-coloring of a graph

In other words given a graph we denote each vertices as v_i and v_j where $i, j < n$. If there exists an edge between v_i and v_j we will denote that edge as $e_{i,j}$. We are supposed to color the graph of all vertices with only three colors (R, G, B) such that no edge should have two vertices of the same color. Below are two common facts about 3-colorable graphs.

- **Fact 1:** If we are given a 3-coloring, permuting the 3 colors (R, G, B) still gives rise to a valid 3-coloring. Ie: Coloring all red vertices blue and coloring all blue vertices red gives a valid 3-coloring.
- **Fact 2:** If the graph is not 3-colorable, then at least one edge has matching colors.

2 Zero-Knowledge Proof for 3-coloring Graphs

Let G be graphs on n vertices and define $V = \{v_1, ..., v_n\}$ be the set of vertices,
$E = \{e_{i,j} : \exists$ edge between $v_i, v_j\}$. On input the graph G is known to both parties. The prover is given a private input in the protocol that is the witness which is a 3-coloring of the graph G. The protocol proceeds as follows.

- **Prover:** Given w a 3-coloring of the graph G. Randomly permute the 3-colors to obtain a new coloring. Utilize a commitment scheme to commit the color of all vertices.

$$\forall i \in [n], c_i = COM(v_i, \text{ color of } v_i)$$

- **Verifier:** Pick edge $e_{i,j} \in E$ and send $e_{i,j}$ to the Prover.
- **Prover:** Open c_i and c_j.
- **Verifier:** Return Accept if $c_i \neq c_j$. Reject otherwise.

An explanation of this protocol is provided. At the first step the Prover will randomly permute the 3-colors to obtain a new coloring. This does not modify the validity of w. We use a committment scheme to hide the coloring of each vertex as a string. Every string is essentially hidden and binded. The Verifier is allowed to select one edge. The Prover opens the committment c_i and c_j and hence the Verifier learns the colors of vertices v_i and v_j. Finally the Verifier checks if the two colors c_i and c_j are different. If so, accept. Else reject. Now lets try to prove this protocol is a zero-knowledge protocol of 3-coloring graphs.

Theorem 1 *The above protocol satisfies completeness, soundness with* $\frac{1}{|E|}$*, and zero-knowledge*

Proof.

Completeness: If witness w provides a valid 3-coloring of the graph G. Then the Prover can commit to the colors such that regardless of what edge the Verifier chooses the Verifier will see that $c_i \neq c_j$ and will return accept.

Soundness: We need to show that if w provides an invalid 3-coloring of G. Then the

$$Prob[\text{ Verifier returns accept}] \leq negl(n)$$

Since w is an invalide 3-coloring of G, then there exists edge $e_{i,j}$ such that $c_i = c_j$. Thus

$$Prob[\text{Verifier returns reject}] \geq Prob[\text{Verifier picks } e_{i,j}] = \frac{1}{|E|}$$

Once we have protocol with soundness of $1/E$ we can just repeat the protocol sequentially to improve the soundness.

By sequential repetition if we repeat the protocol k times and $k >> E$ then

$$Prob[\text{ Verifier returns accept}] \leq (1 - 1/E)^k \leq negl(n)$$

Zero-Knowledge: To prove Zero-Knowledge we construct a simulator S which has the code of V^*, the Verifier, and works as follows.

- **Step 1:** Choose random $e'_{i,j} = (v'_i, v'_j)$ and commit to 2 different random colors for c'_i, c'_j. For all other vertices, v_k where $k \neq i, j$. Let $c_k = 0$, the zero string.
- **Step 2:** Send first message to V^* and get $e_{i,j}$ from V^*.
- **Step 3:** If $e_{i,j} = e'_{i,j}$ open c'_i, c'_j. Else go to Step 1.

Now we need to prove the transcript of simulator S is indistinguishable from the transcript of the real world protocol. Intuitively the only difference is that in the real protocol, all commitments to all the vertices are nicely done. They all are colored whereas in S, most of the vertices have a commitment of a zero string. Next thing to note is that all these other commitments will never be opened. So by the hiding property, all those zero string commitments look identical to the commitments of the vertices in the real protocol.

We will use the hybrid lemma to show that these two transcripts are indistinguishable. Let H_0 be the description of the protocol, H_3 be the description of simulator S.

- H_0 Algorithm S_0 has the correct witness w and code of V^*. S_0 acts as an honest Prover and interacts with V^* which means:
  - **Step 1:** Commit colors of vertices and compute first message honestly with witness w.
  - **Step 2:** Get $e_{i,j}$
  - **Step 3:** Open c_i, c_j and if $c_i \neq c_j$ return accept, else return reject.

  Output the transcript τ_0. τ_0 has the same distribution as in the real protocol.

- H_1 Algorithm S_1 has the correct witness w and code of V^*. S_1 guesses a random edge $e'_{i,j}$. S_1 acts as an honest Prover and interacts with V^* which means:
  - **Step 1:** Commit colors of vertices and compute first message honestly with witness w.
  - **Step 2:** Get $e_{i,j}$
  - **Step 3:** Now if $e_{i,j} \neq e'_{i,j}$ go to Step 1. Else open c_i, c_j and if $c_i \neq c_j$ return accept, else return reject.

  Output the transcript τ_1.

- H_2 Algorithm S_2 has the correct witness w and code of V^*. S_2 guesses a random edge $e'_{i,j}$. S_2 computes the first message using $e'_{i,j}$ which means:
  - **Step 1:** S_2 commits the coloring of every vertices to be zero for all c'_k where $k \neq i, j$. c'_i and c'_j are still computed honestly using w.
  - **Step 2:** Get $e_{i,j}$
  - **Step 3:** Now if $e_{i,j} \neq e'_{i,j}$ go to Step 1. Else open c'_i, c'_j and if $c'_i \neq c'_j$ return accept, else return reject.

  Output the transcript τ_2.

- H_3 Algorithm S is the simulator with code of V^*. S guesses a random edge $e'_{i,j}$.
  - **Step 1:** S commits the coloring of every vertices to be zero for all c'_k where $k \neq i, j$. c'_i and c'_j are computed randomly.
  - **Step 2:** Get $e_{i,j}$
  - **Step 3:** Now if $e_{i,j} \neq e'_{i,j}$ go to Step 1. Else open c'_i, c'_j and if $c'_i \neq c'_j$ return accept, else return reject.

  Output the transcript τ_3.

H_0 is indistinguishable from H_1 as the only difference is that H_1 randomly chooses $e'_{i,j}$ until $e_{i,j}$ is found. Thus the two transcripts have identical distribution.

H_1 is indistinguishable from H_2 from the following lemma and informal proof.

Lemma 2 *Distribution of* τ_1 = *Distribution of* τ_2

The proof follows from the hiding of commitment scheme. The basic idea is that suppose somebody comes along that can distinguish between τ_1 and τ_2. Then they can create a distinguisher algorithm that breaks the hiding idea. As a result all commitments which are not opened can be seen externally by this algorithm and this algorithm would output τ_1 or τ_2, however this would contradict the commitment scheme.

H_2 is indistinguishable from H_3 since the only difference between H_2 and H_3 is that H_3 assigns a random coloring to v_i and v_j instead of utilizing the witness w. However since in the first step we permute the witnessed colors of the vertices, then the witness commits c_i and c_j randomly. Thus the distribution of τ_2 = distribution of τ_3. ■

3 Round Complexity and Efficiency

Lets talk about the round complexity, here we described a basic protocol with three steps but you have to repeat in sequentially many times to reach the soundness that we want. In this section we question how to achieve a zero-knowledge protocol with better round efficiency. What if we repeat the protocol in parallel, rather than sequential. In other words if we have many copies of this basic protocol, $\pi_1, ...\pi_n$ then every copy starts with a random permutation of the witness. In particular what makes this problem unique is that the actions of the Verifier in π_2 could depend on the first message of the Verifier in π_1.We ask if it would still remain zero-knowledge and which is difficult to prove.

4 Fiat-Shamir Transformation

Now we go back to the random algorithm model to construct a zero knowledge protocol. Let $\Sigma_1, \Sigma_2, \Sigma_3$ be the three messages sent between the Prover and the Verifier.

- **Step 1:** The message Σ_1 is sent to the Verifier.
- **Step 2:** The message Σ_2 is sent to the Prover based on the random picking of an edge which is viewed as a string. The edge selected would be $e_{i,j} = r \mod |E|$. Σ_2 is computed as $\Sigma_2 = H(\Sigma_1)$. Where $H(.)$ is a public hash function.
- **Step 3:** Since $H(.)$ is a public hash function, the computation can actually be done by the Prover. Thus the Prover sends $\Sigma_1, H(\Sigma_1), \Sigma_3$.

5 Blockchains and Zero Knowledge Proofs

In this section we look at the utilization of Zero Knowledge protocols in Zero Coin and ZCash. Zero coin is the precursor of Zcash. There is a bulletin board which is part of the public ledger and at any given time, this bulletin board has a bunch of coins that are publicly posted.

$$Coins = \begin{bmatrix} c_1^1 & \dots & c_n^1 \\ \vdots & \dots & \vdots \\ c_1^m & \dots & c_n^m \end{bmatrix}$$

Then we have a list of coins which have already been spent. This spent list is smaller than the bulletin.

$$Spent = \{s_1, ..., s_k\}$$

- **Minting** If you want to convert 1 bitcoin to 1 ZeroCoin, the process works as shown below.
  - **Step 1:** The User computes $c = COM_r(s)$ which is a commitment scheme on the serial number of a bitcoin to be minted. The User sends this c to the miner. Note we ignore the transaction fee.
  - **Step 2:** The Miner burns this c value and will put the c value onto the bulletin board, *coin*.
  - **Step 3:** If user A wishes to spend this coin, and send the coin to B, A just sends (s, r) to B.
- **Coin Collection** If B wishes to collect a coin c, the following steps occur.
  - **Step 1:** B creates a new coin $c' = COM_{r'}(s')$
  - **Step 2:** B sends (s, c') and π, a Zero-Knowledge Protocol with a statement defined as s is an opening of one of the commitments in the bulletin board $Coin$. Note that this is an NP statement.
  - **Step 3:** The Miner runs the following check. It checks if π is a valid Zero-Knowledge Protocol. It then checks if s is not already on the spent list. If so, s is put on the spent list and c' is put on the coin list.

Note that this Minting and Coin Collection is crucial as the anonymity of each transaction is hidden. For example in BitCoin, if A sends a coin to B followed by a second coin, the anonymity of A's transaction is lost since the coin itself is being directly transferred. On the other hand with ZeroCoin, A does not send a coin to B but instead simply sends (s, r). Thus although c is publicly announced on $Coin$, it is impossible to track which coin in $Coin$ is retrieved by B.