CSE 5854: Class 08

Benjamin Fuller

February 13, 2018

1 Building Non-Interactive ZK from scratch

In last class we introduced non-interactive proofs and described the Fiat-Shamir paradigm (sometimes called heuristic). In this class we will build a non-interactive zero knowledge proof system from scratch using a different random resource. Rather than both parties having access to a hash function we assume that both parties have access to a polynomial length random string. This is called a *common reference string* or CRS. Note that a CRS can be created using a random oracle by querying the oracle on increasing location $i, i+1, ...$ and is qualitatively weaker assumption.

2 NIZK for Hamiltonian cycle

Our protocol is for the Hamiltonian cycle problem. The core of this protocol was described by Blum, Micali, and Feldman [BFM88]. Recall that in our interactive protocol for the Hamiltonian cycle the prover committed to a permuted graph and based on the challenge from the verifier either opened the whole graph and the permutation or just a Hamiltonian cycle in the graph.

The insight here is to have the random string dictate the random graphs that the prover would have sent. We will view the CRS as describing a number of commitments to adjacency matrices. The prover (who we assume can break the hiding of the commitments) will check to see if these implicit adjacency matrices describe a Hamiltonian cycle or something else. If the adjacency matrix does not describe an adjacency matrix the prover will show this. When it does the prover will map the cycle onto the matrix containing the adjacency matrix.

For now we talk about the bits contained in CRS. We are really referring to the preimage of a perfectly binding commitment of the string. We view the CRS σ as n^6 perfectly binding commitments of an $n^3 \times n^3$ adjacency matrix H where n is the number of vertices in the graph. Furthermore, we assume that each entry of the matrix is 1 with probability n^{-5}. We say that H is good if it contains a submatrix C that is a Hamiltonian cycle matrix (a single 1 in each row and column with the 1s forming a cycle). Furthermore, the rest of the matrix H is 0. That is, H contains n 1s and the rest of the matrix is 0s.

If H is bad then the prover opens all commitments to the edges of H. The verifier accepts if H is indeed bad. The first thing to check is that a verifier can indeed confirm this. It is easy to ensure that out side of some $n \times n$ submatrix the matrix is 0 and that the submatrix is a Hamiltonian path (checking a

Hamiltonian path is easy if there are no other edges as its only the only way to traverse.

If H is good meaning that it contains a Hamiltonian submatrix (and the rest are 0), then prover computes a permutation π to submatrix C and opens all locations in H except for edges of the permuted graph $\pi(H)$ in C as well as providing π.

Before trying to analyze this protocol lets review some intuition. First, the fact that we are considering a submatrix C of a larger H is entirely to get the soundness probability to work out. Its not crucial for the overall understanding of the protocol.

The second main idea is that when C actually has a Hamiltonian submatrix by opening all of parts of C that aren't edges in $\pi(G)$ we know that $\pi(G)$ can be embedded into a Hamiltonian submatrix. To analyze the protocol we need the following:

1. The probability of H containing the Hamiltonian submatrix is noticeable. (It is $\Omega(n^{-3/2})$, this is shown in section 4.10 of Goldreich.)
2. When H is good it cannot be passed off as bad. (This is clear because the entire matrix is opened and its easy to check). So if a good H is sad to be bad the prover will be caught with probability 1.
3. When H is bad there is no reason to cheat as H can just be opened. So passing a bad H off as good does not increase the probability that P^* can succeed.
4. An understanding of whether P^* can cheat when H is good and claimed to be good H.

We focus on the last fact now. We begin by assuming that H is good. Since π is provided V can compute $\pi(G)$. It can therefore check if all non edges of $\pi(G)$ have been opened. This then means that the Hamiltonian path C is a subgraph of G in the sense that $\pi^{-1}(C) \subseteq G$ (otherwise the verifier would not that at least one of these edges was not opened). Thus if the commitment is perfectly binding when $x \notin L$ the prover is caught with probability 1 when H is good.

Together this allows us to analyze

$$\begin{aligned}
\Pr[P^* \text{ can cheat}] &= \Pr[P^* \text{ can cheat} | H \text{ is good}] \Pr[H \text{ is good}] \\
&\quad + \Pr[P^* \text{ can cheat} | H \text{ is not good}] \Pr[H \text{ is not good}] \\
&= 0 \cdot \Omega(n^{-3/2}) + 1(1 - \Omega(n^{-3/2})) = 1 - \Omega(n^{-3/2}).
\end{aligned}$$

As we've seen before this can be amplified by asking for a large number of proofs (enough to expect good Hs to occur with high probability).

We first describe how to simulate this protocol. Here S is allowed to both create the proof and the CRS σ. We describe how to create σ the description of the corresponding proof should be clear to the reader. First the simulator samples honestly $n^3 \times n^3$ matrix (here we think of them sampling the bits and computing the commitment). If the sampled matrix H is not good then this matrix is placed in the CRS. However, if H is good then S throws out the matrix

H and replaces the submatrix C with a π permutation of their starting graph G. This new matrix is placed in the CRS. Then process is repeated if the protocol is being sequentially composed. First note that the replaced matrix produced by S is not good in that it has too many edges (assuming G has more edges than a Hamiltonian cycle). Then S will honestly open all of the matrices. The only point where S is lying is claiming that the modified matrix is actually good. Its often the case that this S never puts a good matrix in the CRS however assuming a computationally hiding commitment scheme this doesn't effect the view too much (this should be verified).

3 Moving to secrecy

Zero knowledge proofs (including proofs of knowledge and non-interactive proofs) will be crucial tools in ensuring honest behavior of participants in interactive protocols. What we're going to turn to now is how to perform computation of any type in secret. We'll start with a basic situation with two parties P_1 and P_2 where P_1 holds a circuit gate with two inputs (OR, AND, XOR) and P_2 holds two input bits. Note that we can view the gate as P_1's input or we can consider a fixed gate and have P_1 provide one input to the protocol. For the moment we are only going to consider parties who follow the protocol but try and learn information they are not entitled to. The first thing that P_1 is going to do is write the truth table for the circuit as follows:

Input 1	Input 2	Output
0	0	b_{00}
0	1	b_{01}
1	0	b_{10}
1	1	b_{11}

The idea behind the scheme is for P_1 to encrypt each row of this table with keys that correspond to the inputs of P_2. P_2 will then ask for keys according to its input. So the encrypted table is developed as follows:

Input 1	Input 2	Output
r_{10}	r_{20}	b_{00}
r_{10}	r_{21}	b_{01}
r_{11}	r_{20}	b_{10}
r_{11}	r_{21}	b_{11}

This table is then turned into encryptions:

$$\mathsf{Enc}_{r_{10}}(\mathsf{Enc}_{r_{20}}(b_{00}))$$
$$\mathsf{Enc}_{r_{10}}(\mathsf{Enc}_{r_{21}}(b_{01}))$$
$$\mathsf{Enc}_{r_{11}}(\mathsf{Enc}_{r_{20}}(b_{10}))$$
$$\mathsf{Enc}_{r_{11}}(\mathsf{Enc}_{r_{21}}(b_{11}))$$

These four encryptions are delivered to P_2 (in a random order). Now for P_2 to evaluate this computation it suffices for them to get the correct r_{1*} and r_{2*} according to their input. Then P_2 will just try and decrypt each of the four values they are given (note we require it is possible to detect "bad" decryptions). We have two privacy properties we are concerned about:

1. P_2 should only learn information about a single label for each input.
2. P_1 should not learn which input P_2 is evaluating the gate on.

A protocol does this is called *oblivious transfer*. P_2 gets to pick one value without seeing the other and P_1 learns nothing about which value was picked by P_1. This will be the focus of the following class.

References

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 103–112. ACM, 1988.