COM S 6830 – Cryptography November 8, 2011

Lecture 18: Zero-Knowledge proofs - Part 3

Instructor: Rafael Pass *Scribe: Karn Seth*

1 Graph 3-colouring

Recall our ZK-protocol for Graph 3-Colouring, which given a graph $G = (V, E)$, operated as follows:

- The honest prover P has access to a 3-coloring $\vec{\sigma}$ for G. He randomly chooses a permutation $\pi \leftarrow perm\{0, 1, 2\}$ and commits to $\pi(\vec{\sigma})$.
- The honest verifier V then chooses a random (i, j) from E and sends this choice to V.
- P then reveals to V the permuted colours of vertices i and j, namely $\pi(\sigma_i)$ and $\pi(\sigma_j)$ and the randomness r_i and r_j he used to commit these colours.
- V checks if the original commitments match the revealed values, and that the two colours $\pi(\sigma_i)$ and $\pi(\sigma_j)$ are different. V accepts if both these conditions hold.

Completeness of the scheme is easy to see: The honest prover can answer all of the verifier's queries correctly such that the honest verifier will be convinced as long as the commitment scheme is binding.

The scheme also has soundness of $1/|E|$. This follows from the fact that a cheating prover must commit to a colouring that has atleast 1 edge whose two endpoints are coloured with the same colour, and with probability $1/|E|$, the verifier catches him whenever he happens to request the colours for this edge, due to the binding property of the commitment scheme. By performing $n|E|$ sequential repetitions of the protocol, we can get the soundness error down to $(1 - \frac{1}{|E|})^{n|E|} \approx e^{-n}$, which is negligible.

To prove ZK, we will use a simulator very similar to the one used for the Graph Isomorphism problem, operating as follows:

- Given a graph $G = (V, E)$, S chooses an edge (i, j) at random from E. He colours i and j different colours, and colours the rest the same colour, say colour 0. He then permutes the colours randomly, and commits to this permuted colouring, just as the honest prover does.
- He then simulates the verifier V^* on these commited colourings, and receives (i', j'), the first message V^* sends.

- If $(i, j) = (i', j')$, then the simulator can honestly answer the query, and thus simulates the remainder of the protocol, and outputs the transcript.
- If $(i, j) \neq (i', j')$, then the simulator restarts, from the first step, with a fresh (i,j).

Notice that the choice of (i', j') makes must be (i,j) with probability very close to $1/|E|$, since the only information it is given are the commitments, which intuitively cannot bias its decision because of the hiding property. In other words, a V^* that outputs (i, j) with probability very different from $1/|E|$ can be used to break the multi-message hiding property of the commitment scheme.

It follows that the simulator succeeds with probability close to $1/|E|$, and thus has expected number of iterations close to $|E|$, which means it runs in expected poly-time.

We also need to show that the output of the simulator is indistinguishable from the output of an execution of the protocol with an honest prover. The colours are certainly correctly distributed, since they are just two random different colours, and by the blind-man argument of the previous lecture, the edge requests are approximately correctly distribted as well.

However, a subtle difficulty arises in the distribution of commitments, because both the output of the simulator and the view of the verifier working with an honest prover contain a set of commitments out of which two items have been revealed. It may be the case that an adversary with access to the set of partially decommited items could infer something about the items that are not revealed, and use this information to distinguish between the output of the simulator and the view of the verifier working with an honest prover.

In general, we don't know how to prove that our commitment schemes are secure against *selective opening attacks*, namely, attacks where an adversary gets to specify a subset S of commited items to reveal, and then tries to distinguish between the remaining items. However, we can show that when S is of constant size, that is, only a constant number of items are revealed, then our scheme remains secure. This is sufficient to handle the case of our simulator, since we only reveal 2 of the $|V|$ commitments.

2 Zero knowledge under composition

Note that reducing the soundness error of our protocols depended on being able to repeat them multiple times sequentially. We need to argue that a sequential repetition of the protocol preserves zero-knowledge.

The proof of this fact heavily requires a use of z, the string of prior information provided to V in our definition of ZK. We can have this z contain the transcripts of all the messages run so far, and modifying our simulator, which also has access to z, to simulate V^*

as before, except now with access to z. Our earlier argument still applies, and we can show that in each sequential iteration, the zero-knowledge condition holds.

Notice that this z only helps us for sequential repetitions. We do not know how to compose ZK proofs in parallel, and in fact, there are examples of protocols for problems where running the protocol just twice in parallel can reveal the entire witness to the verifier.

3 Graph Hamiltonicity

Our proof of a ZK protocol for the NP-Complete Graph 3-Colouring already gave us a way to give a ZK-proof for any NP language. However, the protocol had the undesirable property that it required a large number of sequential repetitions in order to reduce to soundness error to something negligible. We will now give a ZK-protocol for another NP-complete language, and show that we can acheive negligible soundness error with $\omega(log(n))$ repetitions, and then alter our protocol to get one that requires only $\omega(1)$ repetitions.

The problem we will consider is the Graph Hamiltonicity problem, which is the problem of determining whether a given graph contains a Hamiltonian cycle. A Hamiltonian cycle is a cycle that includes every vertex in the graph. This problem is well known to be NP-complete. We give a ZK-proof for it as follows:

- Given a graph $G = (V, E)$, and a Hamiltonian cycle in the graph $C \subseteq E$, the honest prover P chooses a random permutation $\pi \leftarrow perm\{1, \ldots, |V|\}$, and commits to π. He further commits to the adjacency matrix $G' = \pi(G)$. (A vertex $i \in V$ gets mapped to $\pi(i) \in V'$, and an edge $(i, j) \in E$ gets mapped to an edge $(\pi(i), \pi(j)) \in E'$).
- The honest verifier V then chooses a random $b \in \{0, 1\}$ and sends it to P.
- If $b = 0$, P reveals π and G'. If $b = 1$, P uses $\pi(C)$ to show V a Hamiltonian cycle in G'.
- If $b = 0$, V checks that the information sent by P matches the commitment, and further verifies $G' = \pi(G)$. If $b = 1$, V checks if the cycle P sent is actually a Hamiltonian cycle. If these checks pass, then V accepts.

The completeness of the protocol follows by inspection: an honest prover will always be able to correctly answer an honest verifier's query in a way that causes the verifier to accept.

The soundness of the protocol is 1/2. A cheating prover must either commit to permutation of the original graph, in which case he doesn't know a Hamiltonian cycle in the permuted graph, or he must commit to an unrelated graph where he knows a Hamiltonian cycle, but cannot provide a permutation mapping the original graph to this unrelated graph.

With probability 1/2, the honest verifier requests the information that the cheating prover doesn't have, and thus catches him cheating due to the binding property of the commitment scheme. By performing $\omega(logn)$ sequential repetitions, the soundness error can be reduced to $2^{-\omega(logn)}$, which is negligible.

The simulator for this protocol works similarly to the simulator for the graph 3-colouring problem: it guesses the query the verifier is going to make, and commits to something that allows it to answer that query. Namely, if $b = 0$, it commits to a random permutation of G, and if $b = 1$, it commits to a random permutation of a simple cycle on n vertices.

If the query made by the verifier matches the guess (which it should do with probability $\approx 1/2$, due to the hiding property of the commitment scheme), then the simulator answers the query and outputs the resulting transcript. Otherwise, it restarts.

Since the probability of guessing correctly is $\approx 1/2$, the simulator has an expected number of iterations close to 2, and thus runs in expected poly-time.

By running the protocol $log(n)$ times in parallel, namely, the prover chooses $log(n)$ random permutations of G and commits to them, the verifier makes $log(n)$ 0-1 queries at random, and the prover answers each of the queries using the corresponding permutation of G, we can reduce the soundness error to $1/2^{log(n)} = 1/n$ in each iteration. This is because a cheating prover has to dodge the verifier on each of the $log(n)$ repetitions in order to avoid being caught, and this happens with probability $1/2^{log(n)}$. Further, we only need to run this modified protocol $\omega(1)$ times sequentially in order to make the soundness error negligible.

Notice that the $log(n)$ parallel repetitions doesn't affect the expected polynomial running time of the simulator: the simulator will guess *all* the $log(n)$ queries the verifier is going to ask with probability close to $1/n$, and thus will halt in an expected number of iterations close to n.