CSE 5854: Class 04

Benjamin Fuller

January 25, 2018

1 Building Useful Zero-knowledge

In this reading we will focus on making zero knowledge more useful. Our final goal will be to design zero-knowledge system for all of NP. Recall that we know how to amplify the soundness and completeness of an interactive proof system by sequential repetition. However, it is unclear that this sequential repetition preserves zero-knowledge. Indeed as you see on the problem set sequential composition does not hold if we do not allow the prover and verifier to have access to auxiliary input.

Before we start with these technical results its important to remember why we are defining zero knowledge proofs. The core is that they allow us to enforce honest behavior in a protocol. Suppose that we have two players P_1 and P_2 engaging in an interactive protocol. Suppose that P_1 is sending a message c to P_2. Along with c we have the players engage in an interactive proof that says c would be the next message sent by a player honestly executing the protocol agreed upon by P_1 and P_2.[1] This language is in NP as P_1 could simply show its private information and P_2 could recompute what the next message should be. The important piece is do have this proof be zero-knowledge. If we can achieve this goal we can turn to designing protocols that protect against passive adversaries and then augment them with the appropriate zero-knowledge proofs.

Theorem 1. *Let P be an interactive machine that is* zero-knowledge *with respect to auxiliary input on a language L. Let q be some polynomial and let P_q be an interactive machine that on common input x proceeds in $q(|x|)$ phases each of them running an independent copy of P on input x. Then P_q is zero-knowledge with respective to auxiliary input.*

Furthermore if P is perfect zero-knowledge so is P_q.

Proof. Our goal is to construct a simulator S_q for any malicious verifier V_q^* for the sequentially composed protocol. Recall that we know how to simulate a single interaction for any malicious verifier V^*. Our proof will use the view definition of a simulator. Recall we showed that if a simulator can produce messages computationally indistinguishable from the messages expected by the verifier then the output of the verifier must be computationally indistinguishable. Our composed simulator will do the only thing it knows how to do, it will

[1] There are some subtle issues to deal with here, most pressing is what to do when the parties are supposed to be using randomness. Its not enough to say that the message c would be sent for some random string. You want to guarantee that the party actually sampled a random string.

separately simulate each phase of the protocol. That is, it will independently interact with V_q^* in each round of the protocol.

As a small technical detail we need to partition the malicious verifier V_q^*. We will think of V_q^* outputting everything it has seen at the end of each round, passing this information to the next round of the verifier and then engaging with the prover again. In this way we can think of V_q^* as q separate machines. That is we define $V_0^*, ..., V_{q-1}$ as follows:

1. The machine V_0^* takes an auxiliary input z the auxiliary input of V_q^* and executes the first round of interaction with P. It then outputs its entire view as z_1.

2. The machine V_i^* takes an input z_i and initializes V_q^* running it through $i+1$ rounds of the protocol giving it messages from z_i as appropriate and feeding the random tape from z_i. Once V_q^* has engaged in $i+1$ "rounds" of interaction then V_i^* engages with the real prover. Upon termination, the view is appended to z_i and output as z_{i+1}.

Note that each V_i^* is a polynomial time interactive Turing machine that interacts with P. Thus it follows that there exists a simulator S_i for each of these machines such that,

$$|\Pr[D(x, z, < P, V_i^*(z) > (x)) = 1] - \Pr[D(x, z, S_i(x, z)) = 1] < \mathsf{ngl}(|x|).$$

The full simulator S simply calls each S_i in order creating the new auxiliary input as the output of the previous phase.

Showing that this composed simulator works is a hybrid argument from the fact that each stage is a good simulation. □

Note: This is a much denser proof that it looks at first glance. Make sure you read this proof multiple times and each claim that is being made. Is it really possible to separate the simulator V_q^* as described? Is it possible to strengthen this separation?

Discussion Question 7: Where was auxiliary input crucial in this definition? What fails without auxiliary input?

1.1 ZK for all of NP

Now that we've shown that sequential composition is possible for zero-knowledge we'll show it is possible to build a zero-knowledge (ZK) proof system for all of NP. This will allow us to enforce honest behavior in interactive protocols. Importantly from this point forward we'll assume that the prover has a witness as input and construct protocols where P runs in polynomial time. This will be crucial for actually using ZK in protocols. We'll discuss later how a prover gets access to a witness. Rather than showing an ZK proof system for all of NP we'll show an ZK system for an NP-complete. Sticking with our theme of graph problems we will show that 3-colorability has a zero knowledge proof system.

Definition 1. *Let $G = (V, E)$ be a graph. The graph G is said to be 3-colorable if there exists a permutation $\pi : E \to \{0, 1, 2\}$ such that whenever $u, v \in E$, $\pi(u) \neq \pi(v)$. The graph G is not 3-colorable if no such permutation exists. Define the language $\mathcal{L}$ as the set of all 3-colorable graphs.*

Theorem 2. *Assuming the existence of (information-theoretically binding) commitment schemes* $(\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$*, there exists a zero-knowledge (auxiliary input) interactive proof system* (P, V) *for* $\mathcal{L}$ *(3-colorability). Furthermore, when* P *is given a three-coloring* π *as input* P *runs in polynomial time.*

Proof. We begin by describing the code for P and V. Recall that P has auxiliary input π that is a 3-coloring of the graph G.

1. The prover inputs G, π.
2. The prover samples $ck \leftarrow \mathsf{Setup}(1^{|x|})$. The prover selects a random $\psi : \{0,1,2\} \to \{0,1,2\}$ and computes $\phi = \psi \circ \pi$.
3. For each vertex $u \in V$, P creates $(c_u, d_u) \leftarrow \mathsf{Commit}(\phi(u))$ and sends c_u to the verifier.
4. The verifier randomly selects some edge u, v and sends this edge to P.
5. The prover receives some edge u, v and sends d_u and d_v to V.
6. The verifier computes $(a, b) \leftarrow \mathsf{Open}(c_u, d_u$ and accepts iff $a \neq b$ for both received edges.

□

Discussion Question 8: Why should the commitment binding be information-theoretic?

Discussion Question 9: Provide a sketch of how the simulator should prepare its first message to an arbitrary verifier V^*.