CSE 5854: Class 01

Benjamin Fuller

January 17, 2018

1 Class Arc

As mentioned in class we will spend this semester talking about solving interactive problems using the same techniques covered in CSE 5852. In that class we focused on sending messages over a malicious network in a way that prevented outside interference. One of the main exceptions was key agreement which was an interactive protocol with no message being communicated.

Our final goal is this class will be for a group of people to get together and perform some computation with various privacy goals and maintain these goals even if people participating in the protocol are malicious. So we will no longer consider a "network" adversary that sees communication of honest parties, instead the adversary will be part of the actual computation. This change can make definitions more unwieldy.

2 Motivating Commitments

One of the most basic protocols we will consider is called digital commitments or just commitments. The idea is force someone to make a choice without them having to reveal it. Consider for example two people that wish to play rock, paper, scissors over the internet. This game can be won perfectly if one player is allowed to act "second." Simultaneous moves are crucial for the game's equilibrium. We'd like both players to set their move and not be able to change it and then both players announce their move.

Discussion Question 1 What informal properties do we want out of such a scheme? For the purposes of discussion you can consider two players sender, S, and receiver, R.

Discussion Question 2 Does encryption immediately lead a commitment scheme? Why or why not? Do you need certain conditions?

Discussion Question 3 In previous cryptographic messages, we considered an adversary that receives some information and tries to predict some value (either a forgery or what message was encrypted). What is an adversary trying to do here? What "role" is the adversary playing?

3 Defining Commitments

Unlike previous items, there is no reason to expect that a commitment scheme is a single message. In fact this doesn't really make sense, at a minimum we want

S to send two values separated in time. The first will commit the sender to some value and the second will tell the receiver what value was "committed." For the moment, we will consider an non-interactive version of the problem between two players S and R.

Definition 1. *A commitment scheme for a message space M is a triple of interactive protocols $(Setup, Commit, Open)$ such that:*

1. *The algorithm $ck \leftarrow Setup(1^k)$ generates a public commitment string.*
2. *The algorithm $(c, d) \leftarrow Commit_{ck}(m; r)$ for some message $m \in M$. Here we use the semicolon to indicate that r is randomness of the commitment algorithm.*
3. *The algorithm $M \cup \{\perp\} \leftarrow Open_{ck}(c, d)$ outputs either a message or nothing $\perp$.*

Discussion Question 4 Under this notation, try and formalize your goals above.

3.1 Security Goals

The two standard security goals of a commitment scheme are called binding and hiding. We will give a first definition based on the indistinguishability definition of previous schemes.

Definition 2. *A commitment scheme $(Setup, Commit, Open)$ for M is* hiding *if for all $m_1, m_2 \in M, \forall A \in PPT$,*

$$\begin{aligned} &|\Pr[ck \leftarrow Setup(1^k); c \leftarrow Commit_{ck}(m_1); A(ck, c) = 1] \\ &-\Pr[ck \leftarrow Setup(1^k); c \leftarrow Commit_{ck}(m_2); A(ck, c) = 1]| < \mathsf{ngl}(k) \end{aligned}$$

The second definition is a little more nuanced and is called binding. Here our goal is to prevent the sender from "changing" what value they committed to. Here there's a couple of nuances, first there is no reason to expect that a sender that is trying to cheat is going to run the algorithm *Commit* they may produce arbitrary values. Second, we want to make sure that the sender can't force multiple items to be opened. How do we define this? We want to say that they are locked in once c is seen but maybe there is no possible m at this point. Indeed, it doesn't make sense to talk about what value is contained in c. However, it does make sense to talk about the behavior of *Open*. This gives us the following definition:

Definition 3. *A commitment scheme $(Setup, Commit, Open)$ for M is* binding *if $\forall A \in PPT$,*

$$\begin{aligned} &|\Pr[ck \leftarrow Setup(1^k); (c, d_1, d_2) \leftarrow A(ck); m_1 \leftarrow Open_{ck}(c, d_1) \wedge \\ &\quad m_2 \leftarrow Open_{ck}(c, d_2); m_1 \neq m_2 \wedge m_1 \neq \perp \wedge m_2 \neq \perp]| \leq \mathsf{ngl}(k). \end{aligned}$$

Notice the differences between the definition of hiding and binding. In hiding, we have the *Commit* algorithm being honestly run and an adversary A trying to learn from the output of that algorithm. In binding, we have the *Open* algorithm

being honestly run and an adversary A trying to make that algorithm output two different messages. Although we called an adversary A in both of these settings, it makes more sense to consider an A that is replacing the sender in the protocol for binding and an A that is replacing the receiver in the protocol. That is, the adversary is now playing the roles of one of the parties. This is what we'll consider in the rest of the class, the adversary is not outside of the protocol anymore. Furthermore, this is the first example where we needed multiple different security definitions because the parties were not symmetric in the functionality definition. This means that we needed to consider a different definition based on which role was acting adversarially.

Discussion Question 5 Why don't we consider a definition where both parties are acting adversarially?