Princeton University
Scribe: Jiaxin Guan

Notes for Lecture 14

1 Motivation

Consider a signature scheme described by the game illustrated in figure 1.

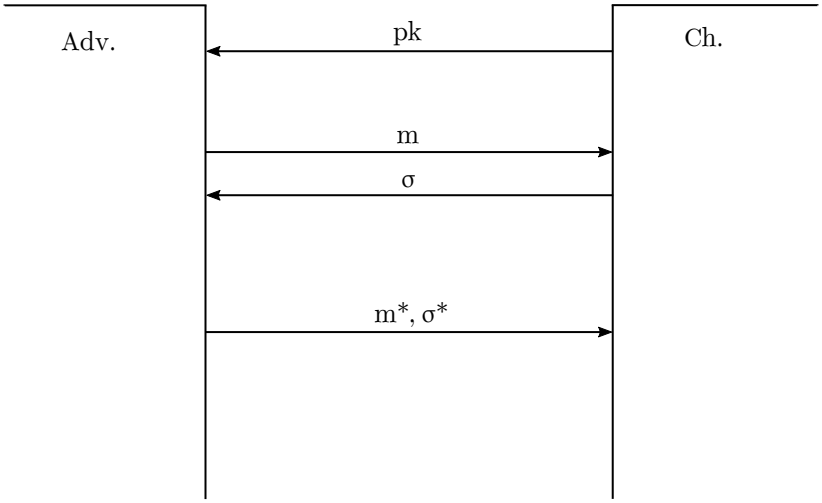

Figure 1: The challenger first sends public key `pk` to the adversary. The adversary then sends message m, and gets the signature σ back. At the end of the experiment, the goal of the adversary is to come up with a forgery m^*, σ^* such that $m^* \neq m$, and σ^* is a valid signature on m^*.

Remember when we talked about the random oracle model, assume there is a signature scheme that uses a hash function which we cannot prove any security properties like collision resistance about. Instead, we can model the hash function as a truly random function, and, in addition, give the adversary oracle access to the truly random function. The game thus becomes the one illustrated in figure 2.

Notice that the challenger now also behaves as an oracle for the truly random function.

We talked about the case where we make the random oracle queries quantum queries, as shown in figure 3.

So now a natural question lies: what if we also switch the signature queries to quantum queries?

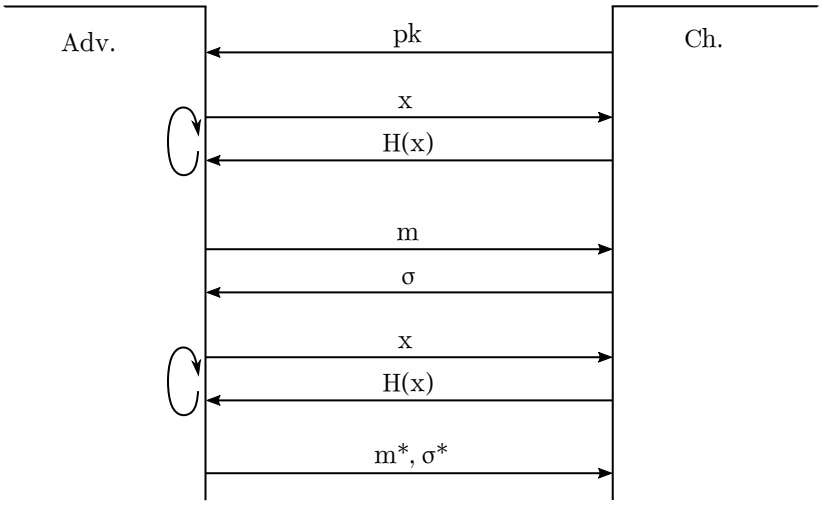

Figure 2: In addition to the orignal communication, the adversary can now repeatedly submit queries x to the challenger, and the challenger would respond with the truly random function H evaluated on x.

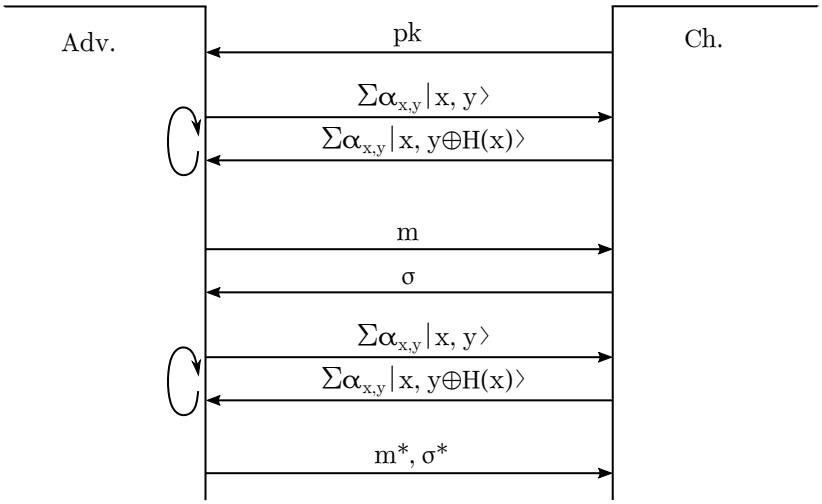

Figure 3: Instead of submitting classical queries x, the adversary now sends superpositions over all inputs $\sum \alpha_{x,y}|x, y\rangle$, and receives as responses superpositions of the outputs $\sum \alpha_{x,y}|x, y \oplus H(x)\rangle$.

Notice that the reason that we make the random oracle queries into quantum queries is because in real life, the adversary knows the code of the hash function. They can thus implement Grover's algorithm, collision finding, etc., i.e. they can interact with the random oracle in a quantum way.

However, for the signature query, it's different: Say Alice is sending a message m and a signature σ to Bob. Here we don't need to worry about quantum queries because

the query corresponds to the message m that Alice sends. And in a classical setting, Alice is also the one generating the message m. There is no reason for Alice to make that a quantum query. Therefore, it's often times sufficient to make random oracle queries quantum, but other queries remaining classical.

Nevertheless, one can be paranoid and be worried that the above claim isn't true. Suppose one have a quantum computer that gets stolen. Then someone could interact with the quantum computer in superposition. Maybe there is some concern here if the adversary is interacting with the quantum computer in superposition.

Another reason to worry about this is that classical crypto can be the building block for quantum protocols. If a scheme is only secure in a classical sense, we cannot use it in the construction of a larger quantum protocol.

One intuitive way is to measure before sending the output. Let's assume that the adversary is somehow able to bypass the measurement step. Actually, measurement could be trickier than we expect. In some sense a measurement is actually entangling with the environment / observer. If the device is to measure what comes in, it is essentially entangling it with the input, and that requires storage. Or if it's entangling with the environment, the adversary could also potentially control the environment. So we will assume that no measurement is allowed.

2 Message Authentication Code (MAC)

We will take a look at one-time MACs.

A MAC consists of two algorithm, $\texttt{Mac}(k, m)$, and $\texttt{Ver}(k, m, \sigma)$. It looks a lot like a signature scheme, except that there is no distinct signing key and verification key - there is just one secret key used for both.

Correctness states that for $\sigma \leftarrow \texttt{Mac}(k, m)$, $\texttt{Ver}(k, m, \sigma)$ outputs 1.

Security is defined below by the game in figure 4.

Classically MACs are very easy to construct: k would be the description of a pairwise independent function. Concretely, let $m \in \mathbb{Z}_p$, $k = (a, b) \in \mathbb{Z}_p$ where a and b are randomly chosen, then we have $\texttt{Mac}((a, b), m) = am+b$. To verify, we simply compute the MAC again and check for equality. Why is this secure for classical queries? See homework 2.

Now also recall from homework that this is not secure if quantum queries are used - one can recover the key, as shown in homework 2. This is what we will try to solve.

The idea is to make the key large enough that as the adversary only learns two values mod p in superposition, there is not enough information in the response from the challenger to encode the whole key. Even if the adversary cannot learn the whole key, there is still a problem - it's not clear what the definition of security should be.

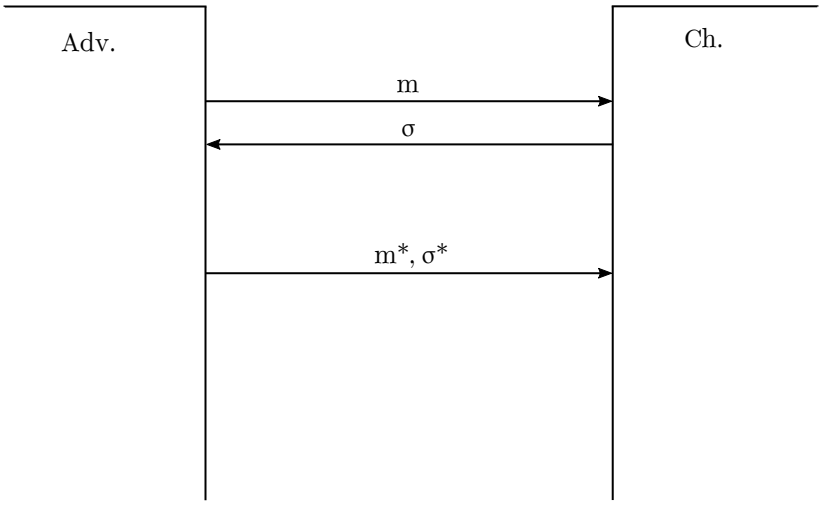

Figure 4: The adversary wins if $m^* \neq m$ and $\texttt{Ver}(k, m^*, \sigma^*) = 1$

Classically, the adversary wins if $m^* \neq m$, and $\texttt{Ver}(k, m^*, \sigma^*) = 1$. But in a quantum setting, there is no m! All we have is a superposition of m. For exmaple, what an adversary can always do is to first send a superposition of all messages, and measure the response from the challenger. The forgery is now a single message while the original query is a superposition over all messages. These are obviously not equal, but obtaining such a forgery is a trivial thing that an adversary can always do. Thus here defining a forgery is not immediately obvious.

One option is that the adversary commits to m^* first before the quantum queries. But that's something along the lines of selective security. We want the adversary to be able to choose m^* at the very end.

Here is how we would modify the game, as shown below in figure 5.

The main idea is that the adversary will now provide two messages together with signatures, instead of a single one. (Notice that we can also extend this definition to allow the adversary to send q queries instead of 1. Consequently, the adversary need to submit $q+1$ message/MAC pairs at the end of the game, as illustrated in figure 6.)

Notice that if we change the queries back to classical, these two games are actually equivalent. The adversary for the latter game can just provide both m and m^* at the very end.

There is still weakness in the definition. We will not talk about the "right" definition today. It's complicated and might be good final project topic.

Here is an intuition of the issue. Suppose the adversary $\mathcal{A}$ gets a superposition of inputs that all starts with "To Alice". Maybe there is some weird way that one can perform some operation on the superposition to get an input that start with "To Bob". Notice that this does not fulfill definition 1, as here the adversary only has

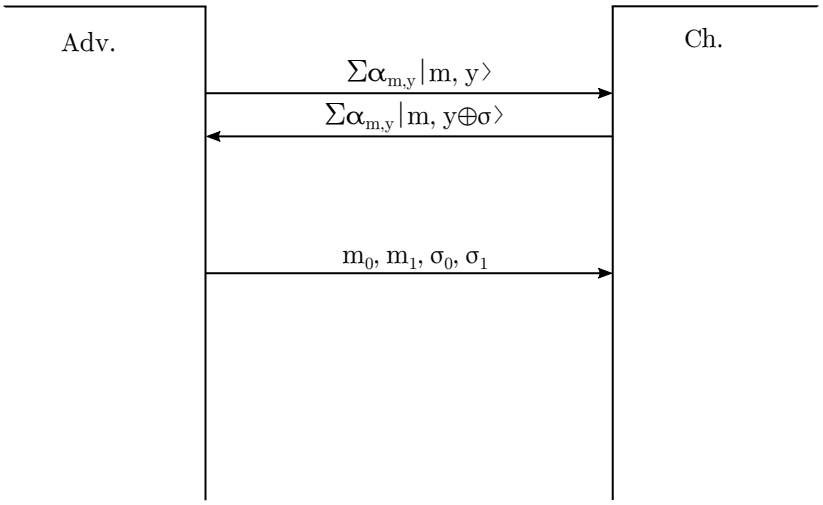

Figure 5: (Boneh-Zhandry '13) The adversary wins if $m_0 \neq m_1$ and $\texttt{Ver}(k, m_b, \sigma_b) = 1$ for $b \in \{0, 1\}$.

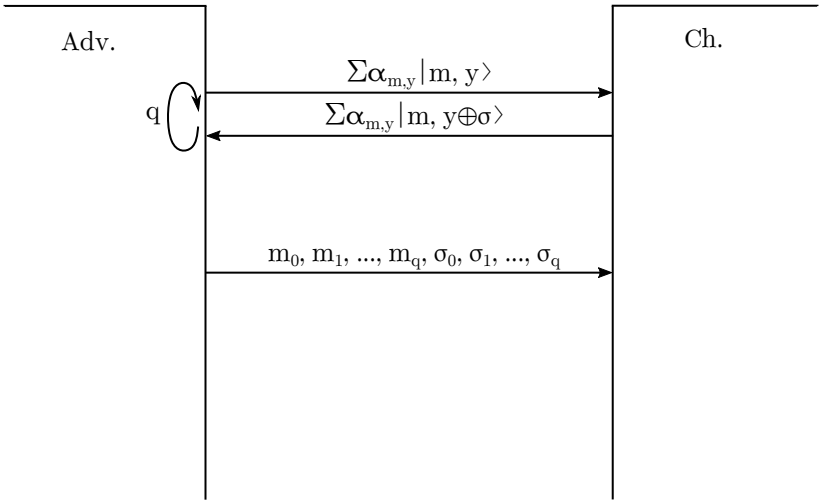

Figure 6: The adversary wins if m_0, $m_1, \ldots, m_q$ are all distinct, and that $\texttt{Ver}(k, m_i, \sigma_i) = 1$ for $i = 0, 1, \ldots, q$.

one forgery. However, this makes sense in the old definition, as the forgery that the adversary outputs looks nothing like the original messages in superposition.

3 Construction

We will first take a look at the classical construction of MACs, and just hope that it works as well in the quantum setting.

Classically, MACs can be constructed using Pseudorandom Functions (PRFs).

Definition 1 (Pseudorandom Function)**.** *A Pseudorandom Function is a deterministic function $F : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$. It takes as input the key $k \in \mathcal{K}$, a message $m \in \mathcal{M}$, and outputs $y = F(k, m) \in \mathcal{Y}$. If k is known, the function F should be easy to compute. Otherwise, the function should look like a truly random function.*

And here is the security definition for a PRF using the security game in figure 7.

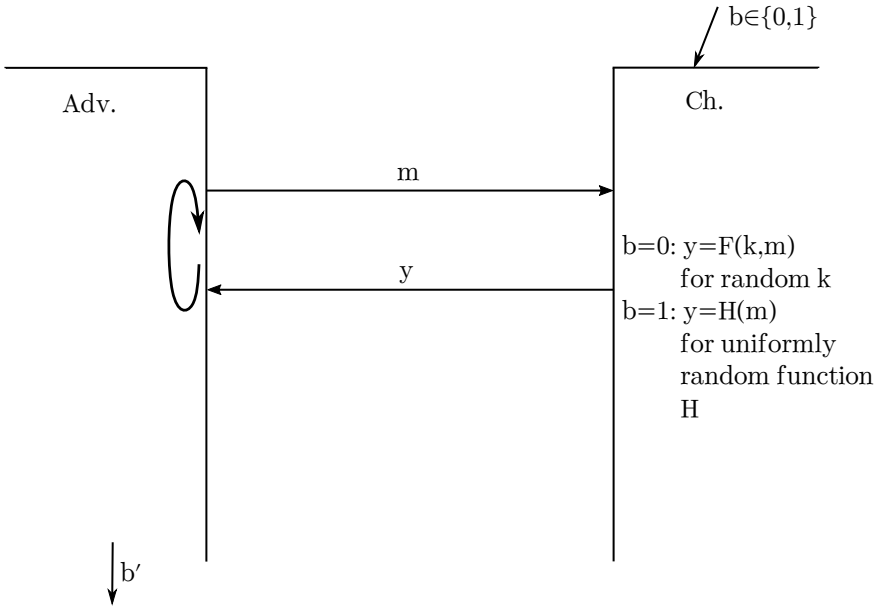

Figure 7: The adversary wins if $b' = b$. (Notice that H and k remain the same throughout the experiment.)

Definition 2 (PRF Security)**.** *F is (t, q, ϵ)-secure if $\forall \mathcal{A}$ running in time $\leq t$ and $\leq q$ queries,*

$$|Pr[b = b'] - \frac{1}{2}| \leq \epsilon.$$

Theorem 3. *(Classical Result) If F is a $(t, q + 1, \epsilon)$-secure PRF and range has size N, then F is a $(t, q, \epsilon + \frac{1}{N})$-secure MAC.*

Here is an intuition for the proof of this theorem: if the function F is uniformly random, then clearly the adversary cannot forge a MAC on a new message. Because the adversary haven't seen the value for the new message, so all the adversary can do is to make a random guess. By security, it is indistinguishable from the case where we have an instance of a PRF instead of the uniformly random function with a loss on the forging probability.

Our idea is to use PRF security to switch to Oracle Interrogation.

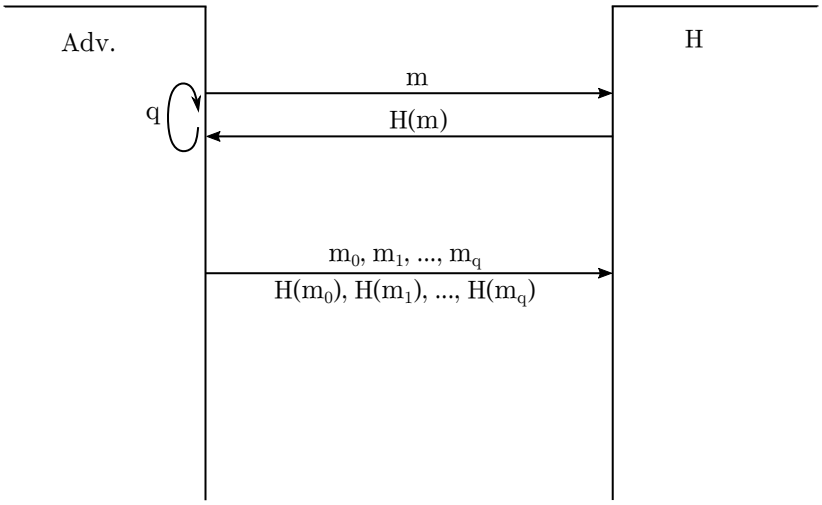

Figure 8: In oracle interrogation, the adversary will be interacting with a random oracle H. The adversary queries the orcale q times and gets back $H(m)$ for each query m. At the end of the game, the adversary is required to output $m_0, \ldots, m_q$ together with $H(m_0), \ldots, H(m_q)$. The adversary wins if all $m_0, \ldots, m_q$ are distinct.

Definition 4 (Orcale Interrogation)**.** *See figure 8.*

Classically, this is hard, because H is uniformly random, and given q queries, if the adversary is outputting $q+1$ points from the function, then at least one of the points must be something that the adversary has received before, and the adversary has no information about it so he cannot do better than a random guess. The question is what happens for quantum? Two things go wrong:

1. The queries for a challenger here are quantum. If we are trying to translate to the oracle interrogation problem, we need to consider quantum oracle interrogation.

2. The other part of the the proof when we switch from MAC forging to Orcale Interrogation is based on PRF security. But we only consinder F secure in a classical setting. Thus we need a quantum-secure PRF.

4 Quantum Oracle Interrogation

In a quantum setting, the adversaries "see" all inputs. They can submit a superposition of all inputs. They are learning information about every input of the function. These inputs are hidden beneath the quantum state so the adversaries can't access them right away. The same argument in the classical setting won't apply, since the adversaries could have information about every one of m_0,...,m_q.

Theorem 5 (von Dom '98)**.** *If $H : \{0,1\}^n \to \{0,1\}$, given q quantum queries, a quantum adversary can get $\approx 2q$ input/output pairs.*

Here the question lies: Is this only applicable because the range is small, or can this theorem be extended to all functions?

Theorem 6. *If $H : \{0,1\}^n \to [N]$, the adversary wins Quantum Oracle Interrogation with probability at most $\frac{q}{N}$.*

5 Quantum-Secure PRFs

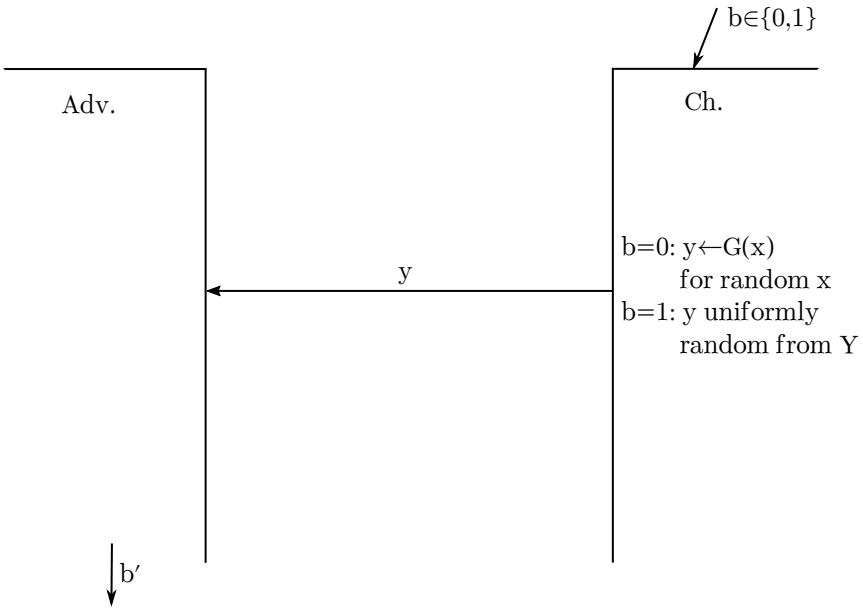

Figure 9: The adversary wins if $b' = b$.

Definition 7 (Pseudo-Random Generator)**.** *A pseudo-random generator (PRG) $G : X \to Y$ with $|Y| >> |X|$ is (t, ϵ)-secure if for all adversary $\mathcal{A}$ for the game illustrated in figure 9 running in time $\leq t$,*

$$|Pr[b = b'] - \frac{1}{2}| \leq \epsilon.$$

Theorem 8 (GGM '84)**.** *PRGs can be used to construct PRFs.*

Before we jump to the construction, let us take a look at the PRF.

Let's assume that the domain of the PRF is small, so small that one can query the entire domain. If one can query the entire domain, then the adversary can construct

the truth table for the function that he is interacting with. The point is that this truth table is either generated by a PRF, or it's a truly random truth table. Thus, in some sense, we can view the PRF as a PRG, which simply evaluates the function F on all inputs and concatenates all the outputs together. So in this case, either the truth table represents one function from the sparse sets of functions, or one that is uniformly random.

Normally, we think of PRFs having relatively large domains. Thus, the truth tables are exponentially big that one cannot learn the whole thing. But the intuition is that PRFs are just very large PRGs.

Here is what we do to use PRG to build a PRF. We repeatedly expand the input using PRGs (assuming that the PRG doubles the input length), as shown in figure 10.

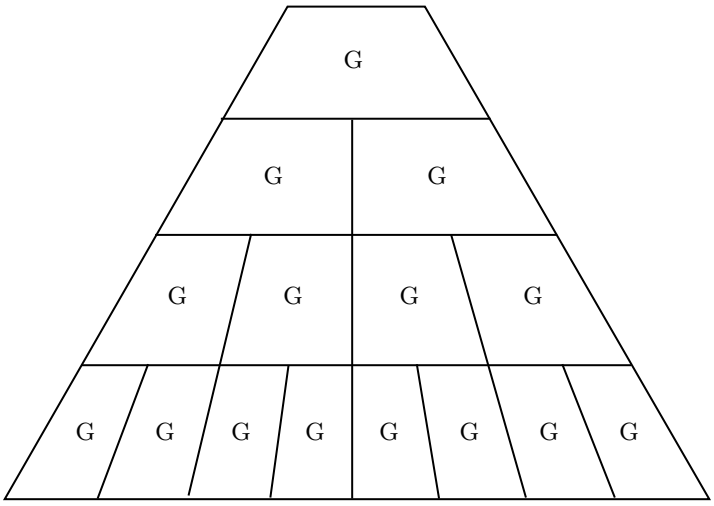

Figure 10: PRF construction from PRG.

The idea is that we're going to do this so many times n. Now the output is 2^n length and one cannot write it down. But if asked for a specific bit, one can easily compute this. One just need to follow down the path from the root of the PRG tree.

Let's say we want to build a PRF $F : \mathcal{K} \times \{0,1\}^n \to \mathcal{Y}$ and that we can write $G(x)$ as $(G_0(x), G_1(x))$. To evaluate $F(k, x)$, we simply compute $G_{x_n}(\ldots G_{x_2}(G_{x_1}(k)))$ where $x_1 x_2 \ldots x_n$ is the bit representation of x.

Classically, this is proved using a hybrid argument. We define the following hybrids:

$$H_0 : F(k, x)$$

$$H_i : G_{x_n}(\ldots G_{x_{i+1}}(H(x_1 \ldots x_i)))$$

and

$$H_n : H(x)$$

where H is a random function. The hybrid argument uses the fact that by triangle inequality, there must be two adjacent hybrids that the adversay should be able to distinguish. We will discuss this further in the next lecture.