Notes for Lecture 15

1 Quantum Security of GGM

(continued from last lecture)

1.1 Classical GGM Proof Recap

In **Hybrid 0** you get oracle access to the function $PRF(k, \cdot)$. In **Hybrid** n you get access to a truly random function. Generally, for **Hybrid** i, you remove the first i layers of the tree and you replace that with random.

If an adversary breaks security of the PRF with some noticeable probability, there must exist i such that the adversary distinguishes **Hybrid** i from **Hybrid** $i+1$ by the triangle inequality. We notice that the adversary can simulate the rest of the layers for itself since they are the same between the two hybrids, so what it's really doing is distinguishing based solely on the switch from PRGs to random in the ith layer.

1.2 Oracle Security

We define the following "oracle security game" where an adversary can send queries x to a challenger. The challenger holds a secret bit b, and responds with random $H(x)$ if $b = 0$, or with $G(R(x))$ where R is a random function. So what we have is an adversary that breaks oracle security of the PRG. We define the game so that the adversary is free to make polynomially many queries.

What we want is that standard PRG security of G implies oracle security. Here's a proof of this in the classical setting.

Proof: Suppose A makes q queries in the oracle security game. We come up with an adversary B that distinguishes q PRG samples from random (recall that standard PRG security only asks that we distinguish one PRG sample from random). All B has to do to simulate the view of the oracle security challenger for A is to forward its ith sample as the response to A's ith oracle query. This simulates A's view perfectly, provided that A only makes distinct oracle queries (otherwise it can answer for itself). So it remains to show that given an adversary B that distinguishes q PRG samples

from random, we come up with an adversary C that distinguishes based on just 1 sample. This last step follows from a hybrid argument.

Which part of this proof breaks when we move to the quantum setting? The difference is that now, the adversary to the oracle security game gets to make quantum queries of the form $\sum \alpha_{x,y}|x, y\rangle$ and gets the response $\sum \alpha_{x,y}|x, H(x) \oplus y\rangle$.

Essentially, all the steps are unchanged except for the step that goes from saying an adversary for oracle queries implies an adversary that distinguishes q PRG samples from random. This step is no longer valid in the quantum setting, since an adversary B that receives q samples cannot simply construct a valid quantum oracle response.

1.3 Fixing the Reduction

So what's the solution? It turns out that we already saw the technique to fix this. Remember in the signature setting we were able to replace a random oracle $H(x)$ with a composition of two random oracles $I(J(x))$ where $J : \chi \to [r]$ and $I : [r] \to \gamma$, and the point was that r was quite small. We proved a theorem that H is indistinguishable from random except with probability $O(q^3/r)$.

Suppose A breaks the oracle security with advantage ϵ. We define the following sequence of hybrids.

- **Hybrid** 0 This is the oracle security game in the random case ($b = 0$), where the adversary receives responses $\sum \alpha_{x,y}|x, H(x) \oplus y\rangle$ for truly random H.
- **Hybrid** 1 Same as before, except now $H(x) = I(J(x))$ for random I and J.
- **Hybrid** 2 Same as before, except now $H(x) = G(I'(J(x))$.
- **Hybrid** 3 Same as before, except now $H(x) = G(R(x))$ (i.e. the $b = 1$ case of the oracle security game)

Hybrid 0 is indistinguishable from **Hybrid** 1 except with probability q^3/r.

Also, **Hybrid** 2 is indistinguishable from **Hybrid** 3 except with probability q^3/r, since we notice it's essentially the same distinguishing problem (with G applied to it, but this cannot possibly increase the distinguishing advantage).

What we conclude is that **Hybrid** 1 and **Hybrid** 2 must be at least $\epsilon - O(q^3/r)$ due to the triangle inequality. So we're breaking oracle security for I versus $G \circ I'$. I corresponds to the random case, and then $G \circ I'$ is the PRG applied to random I' (note that I' has shorter output since it's outputting the seed for G).

At first it might look like this hasn't bought us anything, since we've used an oracle adversary to output another oracle adversary. However, the point is that I and $G \circ I'$

have small domain, namely $[r]$, so now I can have an adversary that queries the entire domain. This allows me to make the reduction work and break the PRG for r samples; note that since r is polynomial I can turn this into an adversary for 1 sample.

Note that in the real reduction, we need to replace the I and I' with $2q$-wise independent functions because we need to be able to simulate them efficiently.

2 Quantum Money

Now we'll switch tracks entirely and consider settings where the honest party also has a quantum computer. One thing that becomes possible is quantum money.

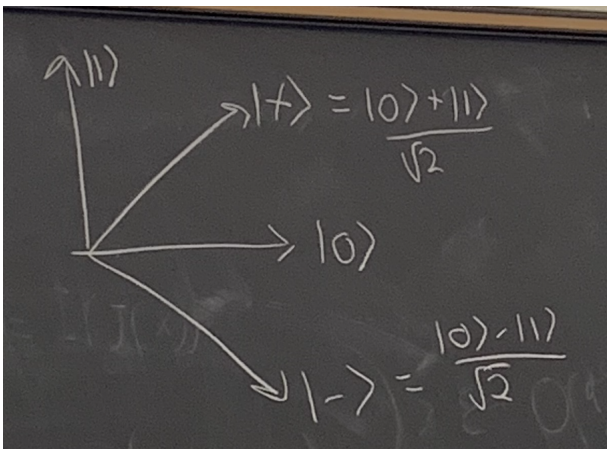
2.1 Quantum Money Basics

As an exercise, let's think about what features we want out of our currency. One of the most important properties is that once I give currency to a merchant, I shouldn't be able to go and spend that bill again.

In the classical digital setting, how do I prevent copying of a string of 0's and 1's? The only way more or less is to have the mint involved in every transaction. Note that blockchains are really just make the mint distributed, but don't remove them from the equation.

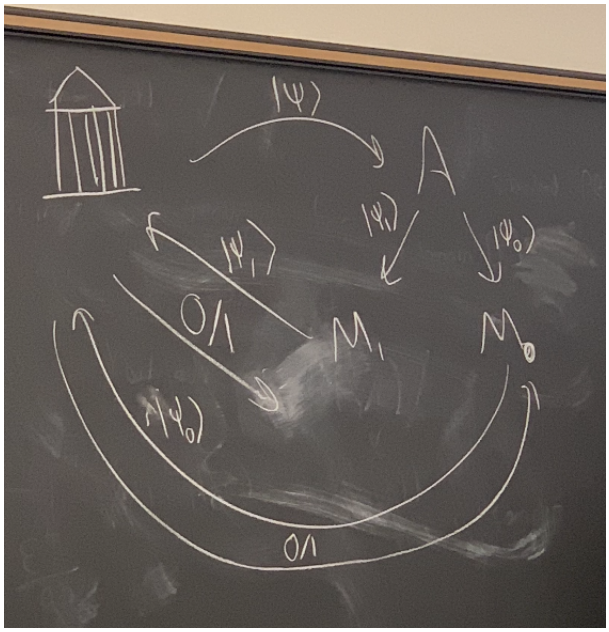
But in the quantum setting, no cloning prevents copying of states. So the idea is that our currency will just be an unknown quantum state, and by no cloning I won't be able to copy it. The way to think about this is that no cloning is really equivalent to quantum money, which is an observation by Weisner in the 60's.

The most basic quantum money algorithm is going to be the following. Imagine we have a mint that only produces a single banknote. The banknote is just going to be a single random choice from the set $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$:



The serial number will just be a 2-bit classical description specifying which one of the four states it is.

There is a mint that produces a quantum state $|\psi\rangle$ and sends it to an adversary A. A will try to come up with two copies $|\psi_0\rangle, |\psi_1\rangle$ of the quantum states and send them to two merchants M_0, M_1. Then each merchant checks with the mint to see whether or not the note is valid:



How does the merchant know if the banknote is valid? The merchant will have a verification procedure $\mathsf{Ver}(\sigma, |\psi\rangle)$ where σ is the serial number specifying the banknote. It simply performs a measurement to check if $|\psi\rangle$ is the right state. If $\sigma \in \{0, 1\}$, then I measure in the standard basis and check if the result matches. So there is always some constant probability that I will reject a state that is not the correct one specified by the serial number. If $\sigma \in \{+, -\}$, then I apply Hadamard first before measuring.

Theorem: The probability that the mint accepts both $|\psi_0\rangle$ and $|\psi_1\rangle$ is some constnat $c < 1$.

The actual banknote is n qubits, so now I have at most a c^n (exponentially small) chance of forging. So we can give out more bank notes by just recording the serial numbers along with id's for each banknote. Furthermore, we can avoid storing a large table by using a PRF to generate the σ's using a secret key k.

We then need the mint to be online, and also for there to be a quantum channel between the merchant and the mint.

Another problem is that the verification oracle allows you to break. The idea is that

if you have a bank note, you can toggle the bits one by one using the verification oracle to learn the underlying σ.

2.2 Public-Key Quantum Money

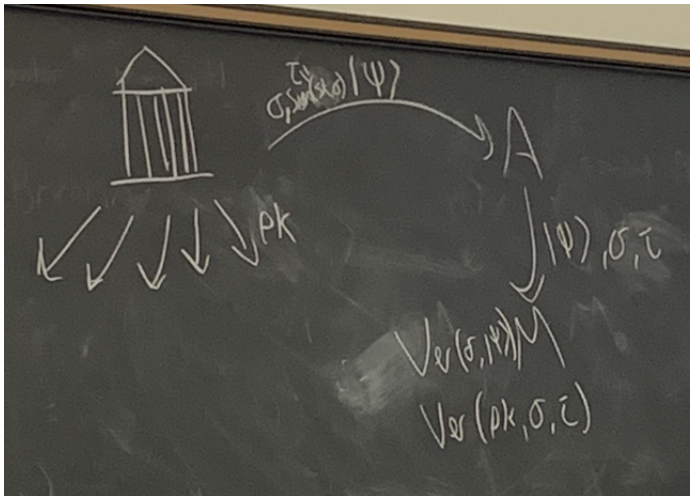
Notice that these two problems stem from the fact that the serial number is secret, and therefore we need the mint to verify. What this brings us to is public key quantum money.

This is a scheme where I can actually maintain security even though the serial number is public. The syntax is as follows:

- $\mathsf{Gen}() \to \sigma, |\psi\rangle$
- $\mathsf{Ver}(\sigma, |\psi\rangle) \to 0/1$.

The definition of security is that if I get a banknote with a serial number, I shouldn't be able to forge another bank note with the same serial number.

So the mint will generate a $\sigma, |\psi\rangle$ and send this banknote to the adversary. Then the mint will broadcast σ, and the merchant can simply use this to run Ver:



I can solve the problem of having to verify with the bank list of serial numbers each time by simply using a classical signature scheme to sign the serial number. Now a bill also contains a signature τ along with $|\psi\rangle, \sigma, \tau$.

So an adversary that tries to make new money will have to either

- increase their supply of serial numbers, which violates signatures security (because this is a forgery)

- create two quantum states for the same serial number, which violates one note security.

In much of the literature on quantum money, this one-bank-note scheme is called a mini scheme.