COS 597A: Quantum Cryptography — Princeton University
Lecture 18 (November 19, 2018)
Lecturer: Mark Zhandry — Scribe: Xin Sun

Notes for Lecture 18

1 Security Of Quantum Lightening

We will begin this lecture with a problem about the proof of security of quantum lightening. Remember from last time that we have some function $H : \{0,1\}^n \to \{0,1\}^n$, where H is collision-resistant but not collapsing. Then the game involving two adversaries: A_0, A_1, and a challenger Ch looks like this:

- A_0 generates the initial state $|\psi\rangle = \sum \alpha_x |x\rangle$ and an internal state, and it passes the internal state to A_1
- Ch receives $|\psi\rangle$ and it flips a bit b:

  If $b = 0$: Measure $|\psi\rangle$

  If $b = 1$: Measure H applied to $|\psi\rangle$

  In the end Ch outputs the state $|\psi'\rangle$ to A_1
- A_1 receives the internal state and $|\psi'\rangle$, and eventually it outputs a bit b'

So in the game the adversary tries to guess whether b' is 0 or 1. Our goal is to come up with a quantum money/lightening scheme, and here is the scheme:

- $Gen()$:

  Run $A_0 \to |\psi\rangle$, $state$

  Measure H applied to $|\psi\rangle \to y$

  Output $(|\psi'\rangle,\ state)$ as the money state, and y as the serial number
- $Ver(y,\ |\psi'\rangle,\ state)$:

  Measure H applied to $|\psi'\rangle \to\ y'$

  Verify if $y = y'$

  Apply A_1 to $|\psi'\rangle$ and $state$, and verify that the outcome is 1

But there is a problem with this scheme: Suppose that the internal state always begins with 0, so now I have a way to cheat by letting my *state* begin with 1, then A_1 will automatically accept. Therefore we don't want to allow arbitrary state.

Here is one way that I can tell whether an adversary is cheating. We are going to replace the third step in $Ver(...)$ above with the following:

- Flip a bit b:

  If $b = 0$: Apply A_1, and check if the result is 1

  If $b = 1$: Measure $|\psi'\rangle$, apply A_1, and check if the result is 0

So if an adversary succeeds with 100% probability, then I know that $|\psi'\rangle$ is in superposition, which is just a random string.

2 Certifiable Randomness

Now we will jump back to certifiable randomness (Notice that in the scheme above, the serial number has to have entropy, otherwise I can have two states with the same serial number). Today we are going to find out how to do verifiable entropy with only classical interactions, meaning that the verifier is also classical, which requires an interaction protocol. The first thing we are going to look at is the building block called Trapdoor 2-to-1 function.

2.1 Trapdoor 2-to-1 Function

$Gen() \to f, f^{-1}$, where f is exactly 2-to-1 and $f^{-1}(y)$ outputs both preimages of y, if y is in the range.

Here is the protocol: We have an adversary A and a client C, where C is classical with small randomness r.

- C runs $Gen() \to f, f^{-1}$, and it gives f to A
- A applies f on the state $\sum_x |x\rangle$ and measures $f(x) \to z$, $|\psi_z\rangle = |x_0\rangle + |x_1\rangle$, where $f(x_0) = f(x_1) = z$, and A gives this z to C
- C flips a bit b and it gives this b to A
- A receives the bit b and:

  If $b = 0$, measure $|\psi_z\rangle$ and gives the result x_c to C

  If $b = 1$, apply $H^{\otimes n}$ and measure, and gives the result ω to C

- For C:

  If $b = 0$, check $f(x_c) = y$

  If $b = 1$, check $\omega \cdot (x_0 \oplus x_1) = 0$, where C gets x_0 and x_1 from f^{-1}

At the end of the protocol, C should get a random string s (statistically random, and therefore pseudorandom is not enough), and the adversary A learns nothing about s.

Somehow we need to guarantee that A chooses the state with some entropy. Our goal is to make sure that A has a superposition of the two preimages. In the protocol above:

- $b = 0$ test is to make sure that A knows at least one of the preimages
- $b = 1$ test is the Fourier Transform, which tests whether the state in is superposition, which shows that there is entropy in z

Now what we know is that if C accepts with 100% probability, then z has to have entropy.

2.2 What Happens Next with The Scheme?

- Repeat many times to improve detection.

  Just like quantum lightening, if A is cheating and it doesn't always succeed, then by repeating many times A will get caught with an overwhelming probability.

- Randomness extraction.

  E.g. A simple randomness extractor:

  seed: pairwise independent function h

  Extractor$(h, x) = h(x)$, where h is compressing and is chosen independently from the source

- PRG (pseudorandom generator) to expand C's randomness

Now we go back to the question about where does the trapdoor 2-to-1 function come from (How can we build this function):

- LWE: Approximately

- Obfuscation (heuristic): Let P_0, P_1 be PRP (pseudorandom permutation)'s.

  $b, x \to P_b(x)$, and we can obfusticate this somehow to get a heuristic trapdoor 2-to-1 function.

Next time we are going to talk about how to encrypt and authenticate quantum states.