Uncloneable Cryptography

Lecturer: Mark Zhandry
Scribes: Fermi Ma and James Bartusek

What is uncloneable cryptography? We'll start with the no-cloning theorem, which says that if I give you an unknown quantum state, you can't always produce two copies of it. While we haven't explicitly seen this in this class before, it's been implicit in much of what we've done so far. For example, when we talked about rewinding, *if* it were possible to clone quantum states, then all the difficulty surrounding collapsing hash functions / quantum rewinding wouldn't exist — we could simply copy the adversary's state and perform classical rewinding, using a fresh copy each time we need to rewind.

Today, we'll see how to use the no-cloning theorem to our advantage to do interesting cryptography.

Theorem 15. *Let S be a set of states such that there exist $|\psi_i\rangle, |\psi_j\rangle \in S$ where $|\psi_i\rangle \neq |\psi_j\rangle$ and $\langle\psi_i|\psi_j\rangle \neq 0$. In other words, not all states in S are orthogonal. Then there does not exist a unitary U such that $U|\psi_i\rangle|0\rangle = |\psi_i\rangle|\psi_i\rangle$ for all $|\psi_i\rangle \in S$.*

The constraint that not all states in S are orthogonal is necessary. If all states in S are orthogonal, then you can clone by first measuring in a basis containing all the elements of S as basis vectors.

Proof. Start by writing

$$\langle\psi_i|\psi_j\rangle = (\langle\psi_i|\langle 0|)(|\psi_j\rangle|0\rangle).$$

Then since unitaries preserve inner products, consider what happens when we apply the alleged cloning unitary U:

$$(\langle\psi_i|\langle 0|)(|\psi_j\rangle|0\rangle) = (\langle\psi_i|\langle 0|U^\dagger)(U|\psi_j\rangle|0\rangle) = (\langle\psi_i|\langle\psi_i|)(|\psi_j\rangle|\psi_j\rangle) = (\langle\psi_i|\psi_j\rangle)^2.$$

But since this equals $\langle\psi_i|\psi_j\rangle$, it must be the case that this inner product is either 0 or 1. This violates our non-orthogonality assumption! □

This gives a rough flavor of no-cloning, but you can prove all sorts of generalizations of this.

Now, let's consider also a "classical" no cloning theorem.

Theorem 16. *Let S be a set of distributions such that $\exists D_1, D_2 \in S$ such that $D_1 \neq D_2$ and D_1 and D_2 have overlapping supports. Then there does not exist a process P such that for all $D \in S$, P maps a single sample $x \leftarrow D_b$ into two independent samples $(y, z) \leftarrow D_b^2$.*

What makes this different from the quantum no cloning theorem? And why don't we use classical no-cloning to build crypto?

The key difference is that you can *test* quantum (pure) states. If you're expecting $|\psi_i\rangle$, you can test whether you're given $|\psi_i\rangle$ by measuring $\{|\psi_i\rangle\langle\psi_i|, \mathbb{I} - |\psi_i\rangle\langle\psi_i|\}$. On the other hand, given a classical sample $x \leftarrow D$, there's no way to test that you got a sample from D.

Quantum Money: Wiesner's Idea

We'll use this ability to test states for our first application: quantum money. Unlike classical banknotes, which can always be copied (in principle), we'll be able to use quantumness to construct banknotes that provably cannot be copied. This origin of this idea is due to [Wie83].

Let $S = \{|0\rangle, |1\rangle, |+\rangle, |-\rangle\}$. Define $|x^\theta\rangle = H^\theta |x\rangle$, where H denotes the Hadamard gate. The state $|x^\theta\rangle$ has an associated serial number (x, θ). For simplicitly, let's also assume that only one banknote is in circulation. To mint a banknote, we'll sample random $x, \theta \leftarrow \{0,1\}$, prepare $|x^\theta\rangle$, and keep (x, θ) private.

How do we verify? Given a claimed banknote $|\psi\rangle$, you measure in the basis θ to get $x' \in \{0,1\}$, and check whether $x = x'$. If this is the actual banknote, this check passes, and any other state will incur some positive probability of error.

This scheme has many problems. We can show that an adversary can't succeed *all* the time, but an adversary can certainly get non-negligible probability of forging. In fact, it's possible to show that there exists a cloner that succeeds with probability $\frac{1}{2} + \frac{1}{2\sqrt{2}} \approx 0.85$. We won't show how that works, but it's easy to see how to launch a $5/8$ attack: choose random $\theta' \in \{0,1\}$, measure in θ' basis to get x' and then output two copies of $(x')^{\theta'}$. If you guess $\theta' = \theta$, you get success probability 1. If $\theta' \neq \theta$, then you measure a random bit, and your resulting pair of states will both pass verification with probability $1/4$.

To reduce the probability of forging, we can use parallel repetition. It turns out that you can show (though it's not super easy) that the probability of cloning n of these random states is $\approx 0.85^n$.

The Verification Problem

A major problem with the proposal so far is that we need the mint to verify banknotes for us. This means that every time you need to transact, you need to send the banknote to the mint. This is certainly inconvenient, and also raises questions about what the mint does after verifying the banknote. It turns out that if the mint always returns banknotes after verification, there are ways to attack the protocol and forge money. Another problem is decoherence. Quantum states are fragile, and you don't want your banknote to evaporate during communication with the bank.

We're going to talk about a version of quantum money that in principle solves all of these problems. Informally, public-key quantum money is money where anyone can verify. Formally, a public key quantum money scheme is a pair of efficient algorithms $(\mathsf{Gen}, \mathsf{Ver})$ such that

- Gen samples a serial number σ together with a money state $|\$\rangle$.
- $\mathsf{Ver}(\sigma, |\$\rangle) \rightarrow$ accept/reject.

Correctness requires that $\Pr[\mathsf{Ver}(\mathsf{Gen}())\ \texttt{accepts}] \geq 1 - \mathsf{negl}$.
Security requires that for all efficient adversaries A,

$$\Pr[\mathsf{Ver}(\sigma, |\$_1\rangle) = 1 \text{ and } \mathsf{Ver}(\sigma, |\$_2\rangle) = 1]] \leq \mathsf{negl}$$

where $(\sigma, |\$\rangle) \leftarrow \mathsf{Gen}()$, $|\$_1\rangle, |\$_2\rangle \leftarrow A(\sigma, |\$\rangle)$.

The crucial difference is that the adversary gets the serial number σ, so this can be treated as a public key.

This solves several problems. We no longer need the mint to verify. We also don't need to worry about what happens if the note is rejected, since if there was anything that we could learn from the verification process, we could have done that ourselves. Finally, we also solved decoherence, since we can keep measuring the quantum money state over and over again.

In Wiesner's scheme, there are no computational constraints. No matter how much computational power you have, you'll never be able to forge Wiesner's scheme (assuming the bank does not return money states after verification). However, you can always break public-key quantum money given unbounded computational power, since you can brute force search for a $|\psi\rangle$ that will pass verification.

This brings us to a key challenge with constructing public-key quantum money: given σ, no-cloning no longer holds information-theoretically. So how do we go about proving that a quantum money scheme is secure?

Here's an attempt: let $H : \{0,1\}^{2n} \to \{0,1\}^n$ be a collision-resistant function. Let $\sigma \in \{0,1\}^n$ and let $|\$_\sigma\rangle \propto \sum_{x:H(x)=\sigma} |x\rangle$. Gen() works by constructing $|\psi\rangle = \sum_{x\in\{0,1\}^{2n}} |x\rangle$, and then measuring H applied to $|\psi\rangle$ to get σ. Then $|\psi\rangle$ collapses to $|\$_\sigma\rangle$. We can actually prove that this money state is unforgeable: Suppose that $A(\sigma, |\$_\sigma\rangle)$ outputs $|\$_\sigma\rangle, |\$_\sigma\rangle$, and we measure both copies. Then we can get x_1 and x_2. We know that $H(x_1) = H(x_2) = \sigma$. We also know that $x_1 \neq x_2$ w.h.p., since on average $|\psi\rangle$ is a superposition of 2^n pre-images, so when we measure, we get a collision with overwhelming probability.

The problem is verification. One idea for verification is to just evaluate H on your state and test that the output is σ. But that's not enough. Consider an attack where you measure $|\$_\sigma\rangle$ to get x, and you output $|x\rangle |x\rangle$. In other words, this verification procedure does accept the correct state $|\$_\sigma\rangle$, but it *also* accepts many imposter states that are themselves easy to clone.

There's actually an interesting connection to collapsing hash functions. Recall that a collapsing hash function says that measuring the input is indistinguishable from measuring the output. If the hash function is also collapsing, then there actually *doesn't exist* any (efficient) verification procedure. Indeed, collapsing implies that $|x\rangle |x\rangle$ is indistinguishable from $|\$_\sigma\rangle |\$_\sigma\rangle$, and the latter should pass verification while the former shouldn't.

Public-key Quantum Money from Obfuscation

The origins of this idea are due to Aaronson and Christiano [AC12].

The serial number in [AC12] is $\sigma = S$, a subspace of $\mathbb{Z}_2^n$ of dimension $n/2$. The banknote is $|\$_S\rangle = \frac{1}{2^{n/4}} \sum_{x\in S} |x\rangle$. To verify, we can first check if the support of the state is in S. Then, we apply the Quantum Fourier Transform, and check if the support of the resulting state is in $S^\perp$, where $S^\perp$ is the dimension $n/2$ subspace consisting of all vectors orthogonal to S.

It can be shown that the only state that passes verification is $|\$_S\rangle$. [AC12] observed that to run verification, we don't necessarily need to know S. That is, it seems possible that we could check membership in S and $S^\perp$ without having to give out a description of S in the clear.

[AC12] gave a candidate proposal for how to check membership in S, but it turned out their proposal was broken. On the other hand, they proved that their scheme is secure if the adversary

only has *oracle access* to the programs for S and $S^\perp$.

So, what we need is a real-world instantiation of these oracles. In other words, we need a way to *obfuscate* these programs. In particular, we will want our obfuscator to satisfy the following properties.

Definition 17. *A* subspace-hiding obfuscator Obf *takes as input the description of a subspace S and outputs an obfuscated program $\hat{S}$. It should satisfy the following properties.*

- *Correctness: For all $x \in S$, $\widehat{S}(x) = 1$, and $\forall x \notin S$, $\widehat{S}(x) = 0$.*
- *Security: The following two experiments are indistinguishable:*
  - *Choose random S of dimension $n/2$, and output $S, \mathsf{Obf}(S)$.*
  - *Choose random S of dimension $n/2$, sample random T of dimension $3n/4$ that contains S, and output $S, \mathsf{Obf}(T)$.*

Putting it all together, we get the following proposal for a quantum money scheme, due to [Zha19].

- $\mathsf{Gen}()$: sample $S \subset \mathbb{Z}_2^n$ of dimension $n/2$. Output the public key $\sigma = (\widehat{S}, \widehat{S^\perp})$ where $\widehat{S} = \mathsf{Obf}(S)$ and $\widehat{S}^\perp = \mathsf{Obf}(S^\perp)$ and the money state $|\$_\sigma\rangle = \frac{1}{2^{n/4}} \sum_{x \in S} |x\rangle$.
- $\mathsf{Ver}(\sigma, |\psi\rangle)$: check that $\widehat{S}(|\psi\rangle) = 1$ and $\widehat{S}^\perp(\mathsf{QFT}\,|\psi\rangle) = 1$.

Now we'll prove security.

- Hybrid 0: $\mathsf{Gen}() \to (\sigma, |\$_\sigma\rangle)$. Run the adversary $A(\sigma, |\$_\sigma\rangle) \to |\$_1\rangle\,|\$_2\rangle$. Then run $\mathsf{Ver}(\sigma, |\$_1\rangle), \mathsf{Ver}(\sigma, |\$_2\rangle)$.
- Hybrid 1: Replace the verification algorithm with a projection onto $|\$_\sigma\rangle$. By the correctness of the subspace-hiding obfuscator, this is equivalent to implementing the original verification procedure (projectively).
- Hybrid 2: Choose random $T \supseteq S$ of dimension $3n/4$ and $U \subseteq S$ of dimension $n/4$. Replace $\widehat{S}$ with $\widehat{T} = O(T)$, and $\widehat{S}^\perp$ with $\widehat{U}^\perp = O(U^\perp)$. However, we still verify by projecting onto $|\$_S\rangle$. This is indistinguishable from Hybrid 1 by the security of the subspace-hiding obfuscator.

We claim that the adversary's probability of cloning in Hybrid 2 is negligible. The idea is to consider an equivalent way to sample things. We could have *first* sampled U, T such that $\dim(U) = n/4$ and $\dim(T) = 3n/4$, and $U \subseteq T$. *Then*, we sample random $n/2$ dimensional S such that $U \subseteq S \subseteq T$. One can show information-theoretically, via an appropriate generalization of the no-cloning theorem, that even given U and T, it is hard to map $|\$_S\rangle \to |\$_S\rangle\,|\$_S\rangle$. Alternatively, this follows from the analysis in [AC12], since we are just sampling a random S from the ambient space "between" U and T.

Unfortunately, we currently only know one way to build subspace-hiding obfuscation, which is to use the heavy hammer of general purpose (post-quantum) indistinguishability obfuscation. Obtaining public-key quantum money from other assumptions is a major open problem.

Uncloneable Crypto

There is a whole world of unclonability.

- Aaronson [Aar11] proposed a notion called *copy protection*. The idea is that you have a piece of software that you want to give out, but you want to prevent users from being able to then give out the program to their friends. This is in some sense an obfuscator (in that it takes programs as inputs), but it outputs a quantum state (rather than a classical obfusacted program). It is now known how to copy protect programs using idealized (classical) oracles.
- There are types of uncloneable crypto that don't seem to imply quantum money. For example, in *uncloneable encryption* [BL20], we imagine that the ciphertext is a quantum state, and the adversary shouldn't be able to output two quantum ciphertexts that can both be used to predict the underlying plaintext given the secret key.
- Another example is *secure software leasing* [AL21]: you want to be able to give out a program, and then later *verify* that the user returned the program. If this verification passes, you can be sure that the user doesn't have the program anymore.
- People also study stronger versions of quantum money, such as *quantum lightning* [Zha19] where even the mint can't clone.
- Finally, there is a notion of *one-shot signatures* [AGKZ20]: A signature with a quantum secret key $|\mathsf{sk}\rangle$ and a classical public key pk. The security definition is that it's impossible to obtain $(\mathsf{pk}, m_1, \sigma_1, m_2, \sigma_2)$ where $\mathsf{Ver}(\mathsf{pk}, m_1, \sigma_1) = 1$ and $\mathsf{Ver}(\mathsf{pk}, m_2, \sigma_2) = 1$. It turns out that if you have this object, you can build quantum money with classical communication. The serial number is just the public key $\sigma = \mathsf{pk}$, and the banknote is $|\$\rangle = |\mathsf{sk}\rangle$, generated by Alice. Bob generates his own $\mathsf{pk}', |\mathsf{sk}'\rangle \leftarrow \mathsf{Gen}()$. Then Bob sends pk' over to Alice, and Alice replies with the signature $s = \sigma(|\mathsf{sk}\rangle, \mathsf{pk}')$. So then Bob has the money state $(\mathsf{pk}', s, |\mathsf{sk}'\rangle)$.