Notes for Lecture 12

Let us pick up what we left off. In the last time, we are looking at random oracle model. For some application, we do not know how to prove the security under random oracle model in the quantum setting. One example is digital signature.

Let us recap what is digital signature scheme. There are three polynomial algorithms:

- Gen takes a security parameter 1^λ and outputs a public key pk and a private key sk;
- Sign takes a private key sk and a message m, it outputs a signature σ;
- Ver takes a public key pk, a message m and its signature σ, it outputs 0/1.

The correctness of a signature scheme is defined as follows: for any message m,

$$\Pr\left[\mathsf{Ver}(\mathsf{pk}, m, \sigma) = 1 \middle| \begin{smallmatrix}(\mathsf{pk},\mathsf{sk})\leftarrow\mathsf{Gen}(1^\lambda)\\ \sigma\leftarrow\mathsf{Sign}(\mathsf{sk},m)\end{smallmatrix}\right] = 1$$

We say the digital signature scheme is (t, q, ϵ)-secure if for every adversary $\mathcal{A}$ with running time at most t, A making at most q queries wins the following game with probability at most ϵ.

1. The challenger generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and it broadcasts the public key pk;
2. $\mathcal{A}$ sends message m_i to the challenger and gets the signature $\sigma_i = \mathsf{Sign}(\mathsf{sk}, m_i)$;
3. Finally it is the challenge stage where $\mathcal{A}$ comes up with a pair of m^*, σ^* such that m^* is not in $\{m_i\}_{i=1}^q$. The result of this game is $\mathsf{Ver}(\mathsf{pk}, m^*, \sigma^*)$.

We will first see how to build a signature in the classic world and see what is going to be wrong in the quantum world and how to solve the problem.

1 Digital Signature in the classic world

One of the building blocks in cryptography is Trapdoor Permutation (**TDP**).

- It is going to have a Gen algorithm which produces a pair of keys pk, sk;
- A keyed function $f(\mathsf{pk}, \cdot)$ where the first input is to determine the function and the second input is the "true" input of the function $f_{\mathsf{pk}}(\cdot) = f(\mathsf{pk}, \cdot)$
- An inverse function which takes sk, allows you to go backward: $f^{-1}(\mathsf{sk}, f(\mathsf{pk}, x)) = x$ for all pk, sk and x.

We can think of the function as a permutation which maps n-bit strings to n-bit strings. For security, we have the following, we say it is (t, ϵ)-secure if for any adversary $\mathcal{A}$ with running time at most t,

$$\Pr[f(\mathsf{pk}, \mathcal{A}(\mathsf{pk}, y)) = y] \leq \epsilon$$

The randomness is taken over Gen and the uniform choice of y In other words, an adversary does not have too much advantage to invert $f(\mathsf{pk}, .)$ if it only knows pk. We know how to build it from factoring but it failed as we know Shor's algorithm breaks it.

Here is a really simple scheme built from trapdoor permutation.

- $\mathsf{Sign}(\mathsf{sk}, m) = f^{-1}(\mathsf{sk}, m)$;
- $\mathsf{Ver}(\mathsf{pk}, m, \sigma) = [f(\mathsf{pk}, \sigma) = m]$;

This scheme seems to work according to the security of TDP. But there is an attack that one can simply choose any σ and compute $m = f(\mathsf{pk}, \sigma)$.

An easy modification can makes it work, by introducing a hash function H: $\mathsf{Sign}(\mathsf{sk}, m) = f^{-1}(\mathsf{sk}, H(m))$.

But we do not know how to prove it. This is where the ROM methodology comes in. If we teat H as a random oracle, we can prove the scheme works.

Easy case

Let us consider an easy case where no signing query appears in the security game. Here is the game:

1. The challenger generates $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ and it broadcasts the public key pk;
2. $\mathcal{A}$ can only make oracle queries: it sends m and receives $H(m)$;
3. Finally it is the challenge stage where $\mathcal{A}$ comes up with a pair of m^*, σ^* such that m^* is not in $\{m_i\}_{i=1}^q$. The result of this game is $\mathsf{Ver}(\mathsf{pk}, m^*, \sigma^*)$.

The adversary only gets pk and can make polynomial number of oracle queries. Finally $\mathcal{A}$ should forge a valid pair of message and signature. Assume there is an $\mathcal{A}$ that breaks the security game. We can build an adversary $\mathcal{B}$ that breaks TDP using $\mathcal{A}$ as a subroutine.

- Upon $\mathcal{B}$ gets pk and y, $\mathcal{B}$ gives pk to $\mathcal{A}$ ($\mathcal{B}$ plays the role of the challenger in $\mathcal{A}$'s game);
- If $\mathcal{B}$ gets the challenge query (m^*, σ^*), $\mathcal{B}$ can use the pair as finding a pre-image of TDP $f(\mathsf{pk}, \cdot)$ (will see later);
- How about random oracle queries? What $\mathcal{B}$ is going to do? $\mathcal{B}$ is going to simulate $\mathcal{A}$ on the fly. Every time $\mathcal{B}$ gets an oracle query m from $\mathcal{A}$, $\mathcal{B}$ simulates a random oracle: keep an input/output table, if m is in the table, output the corresponding r; otherwise choose a random r, add (m, r) to the table and return r.

Assume m^* is also an oracle query at the very beginning, we can put (m^*, y) in the table. Because y is random, from $\mathcal{A}$'s view, this is identical to the case where y is chosen uniformly at random. So σ^* returned by $\mathcal{A}$ at the challenger stage is indeed a pre-image of y.

But what if we do not know when m^* is asked (or even m^* is never asked). We can argue that if m^* is never asked, the probability of guess $H(m^*)$ right is negligible. We then randomly guess the i-th query is m^*. So the probability of breaking TDP is ϵ/t.

Hard case

In this case, we allow an adversary to make signing queries. r in the database is now not generated as a random string, but we generate random s first, and let $r = f(\mathsf{pk}, s)$. From $\mathcal{A}$'s view, they are identical. If $\mathcal{A}$ is making a signing query, $\mathcal{B}$ can respond with s instead of r.

With ROM, we can

1. know what $\mathcal{A}$ is doing with the hash function H;
2. know how to embed y into $\mathcal{A}$'s game. We somehow 'program' y into H which $\mathcal{A}$ interacts with.

So with the simulated random oracle, we can still answer oracle queries. Moreover, we can answer signing queries and program the random oracle such that it outputs y on x^*. The previous proof still works with the probability of breaking TDP is ϵ/t.

2 Digital Signature in the quantum world

Now let us switch to quantum world. $\mathcal{A}$ is still getting pk from the challenger. Now because the adversary has the function H, the only way to model that correctly is to allow the adversary to query on superposition. Because an adversary can always get the code of H and implement it as a quantum gate. So we should treat H as a quantum oracle. That is if $\mathcal{A}$ sends an oracle query $\sum_{x,z} \alpha_{x,z}|x,z\rangle$, it gets $\sum_{x,z} \alpha_{x,z}|x,z\oplus H(x)\rangle$. It seems hard to simulate it on the fly.

In the classic setting, we can maintain an input/output table that each entry is chosen uniformly and consistent. But in the quantum world, we have two problems. Considering the oracle query is $\sum_x |x,0\rangle$, the resulting state is $\sum_x |x,H(x)\rangle$ which requires knowing every value of H. If we still keep the database, we need to write down every value of the oracle. And also, it is hard to insert y into a random query point. The success probability is negligible (if we follow the proof for classic setting).

The first problem seems not super hard. If $\mathcal{A}$ makes t queries to H, the followings are indistinguishable (refers to the readings):

- H is a random function;
- H is a $2t$-wise independent function;

We say H is k-wise independent function if for any distinct $x_1 \cdots x_k$ and all $y_1 \cdots y_k$, $Pr[H(x_i) = y_i \ \forall i] = (1/2^n)^k$. In other words, if we only look at k inputs of the function, the output is uniformly at random.

They are easy to construct, here is one example: if H is defined on $\mathbb{Z}_p \to \mathbb{Z}_p$, we can define $H(x) = \sum_{i=0}^{k-1} a_i x^i$ for random $a_0, \cdots, a_{k-1}$. So to write down the polynomial, it is only required to write down H from a $2t$-wise independent function family which solves the first problem.

The second problem: We need to somehow come up with an H that embed y into H and the adversary touches y with high probability. We choose a parameter ℓ, choose $P : M \to [\ell]$ and $Q : [\ell] \to \{0,1\}^n$ be both random function and let $H(m) = Q(P(m))$. We want $Q \circ P$ is approximation of a random function $M \to \{0,1\}^n$.

Classically it is true. We can not distinguish unless find collision for P. The probability of finding a collision is bounded by $O(t^2/\ell)$ so by choosing modestly big ℓ, we will make the probability small. And then we can embed y into Q. And the probability of the challenge query corresponding to y will be divided by ℓ.

- P: t-wise independent random function ($2t$ for quantum);
- Q: random function from $[\ell]$ to $\{0,1\}^n$ but choose a random i and make that entry be y, other entries be $f(pk,s)$ (choose s randomly and compute $f(pk,s)$).

Probability of $\mathcal{B}$ succeeding relies on the following: (1) $\mathcal{A}$ succeeds, (2) $\mathcal{A}$ distinguishes the small range distribution, (3) we guess the index i correctly;

$$\Pr[\text{succeeds}] \geq (\epsilon - t^2/\ell)/\ell$$

To maximize the probability, choose ℓ to be $t^2/(2\epsilon)$ where the probability is $\epsilon^2/(4t^2)$.