The Theory Behind Blockchains (Spring 19)
Recitation 2

Eliad Tsfadia

1 Signature Schemes

1.1 Definition (Reminder)

A digital signature scheme consists of the efficient functions (Gen, Sign, Vrfy) such that

- $\mathrm{Gen}(1^n)$: outputs a pair of keys $(pk, sk) \in \{0,1\}^* \times \{0,1\}^*$.
- $\mathrm{Sign}_{sk}(m)$: output a "signature" $\sigma \in \{0,1\}^*$.
- $\mathrm{Vrfy}_{pk}(m, \sigma)$: output 1 (YES) or 0 (NO).

Properties:

1. **Consistency:** For every $(pk, sk) \in \mathrm{Supp}(\mathrm{Gen}(1^n))$, if $\sigma = \mathrm{Sign}_{sk}(m)$ then $\mathrm{Vrfy}_{pk}(m, \sigma) = 1$.
2. **Existential Unforgability:** For every PPT algorithm A,

$$\Pr_{(pk,sk)\leftarrow \mathrm{Gen}(1^n)}[\mathrm{A}^{\mathrm{Sign}_{sk}}(1^n, pk) = (m, \sigma) \text{ s.t. } \mathrm{Vrfy}_{pk}(m,\sigma) = 1 \text{ and } \mathrm{Sign}_{sk} \text{ didn't query } m] \leq \mathrm{negl}(n).$$

In class you've heard that in theory you can construct signatures from any function that is one-way. However, the general construction is quite complicated. Today we will see a simpler construction which is based on the existence of trapdoor permutation which is unknown to be implied by one-way function, but is very popular in practice (e.g., RSA).

1.2 Trapdoor Permutation (TDP)

Definition 1 (Trapdoor Permutation (TDP))**.** *A trapdoor permutation is given by polynomial time algorithms (Gen, F, F^{-1}) with the following properties:*

- *$Gen(1^n)$ is a probabilistic algorithm that given a security parameter 1^n outputs a secret key (also called the trapdoor) and a public key (pk, sk).*
- *$F_{pk}\colon \{0,1\}^n \to \{0,1\}^n$ is an efficient deterministic algorithm that given a public key pk and input $x \in \{0,1\}^n$, outputs an image $y = F_{pk}(x)$, and F_{pk} is a permutation.*
- *$F^{-1}_{sk}\colon \{0,1\}^n \to \{0,1\}^n$ is an efficient deterministic algorithm that given a secret key sk and $y \in \{0,1\}^n$, outputs $x = F^{-1}_{pk}(y)$.*

Properties:

1. ***Correctness:*** *For every $(pk, sk) \in \mathrm{Supp}(\mathrm{Gen}(1^n))$ and any $x \in \{0,1\}^n$ it holds that*

$$F^{-1}_{sk}(F_{pk}(x)) = x$$

2. ***One Wayness:*** *For every PPT algorithm* A*,*

$$\Pr_{(pk,sk)\leftarrow \mathrm{Gen}(1^n), x\leftarrow\{0,1\}^n}[\mathrm{A}(pk, F_{pk}(x)) = x] \leq \mathrm{negl}(n). \quad (1)$$

A real life example for TDP is RSA: $\mathrm{Gen}(1^n)$ chooses n-bit length primes p and q, and choose an integer $e \in \mathbb{Z}_N^*$ and compute $d = e^{-1} \bmod \phi(N)$ (where $\phi(N)$ is the number of integers in $[N-1]$ that are relatively prime to N), and returns $(pk = (N,e), sk = d)$. For any $x \in [N-1]$, $F_{pk}(x) = x^e \bmod N$, and for any $y \in [N-1]$, $F_{sk}^{-1}(y) = y^d \bmod N$.

1.2.1 Constructing a Signature Scheme from TDP

Assume we are given a TDP triple (Gen_T, F, F^{-1}).

First Attempt

1. $\mathrm{Gen}(1^n)$: Sample $(pk, sk) \leftarrow Gen_T(1^n)$.
2. $\mathrm{Sign}_{sk}(m)$: Output $\sigma = F_{sk}^{-1}(m)$.
3. $\mathrm{Vrfy}_{pk}(m,\sigma)$: Output 1 $\iff F_{pk}(\sigma) = m$.

Note that this scheme is not good enough. First, it doesn't support arbitrary length messages (but lets ignore this issue). More importantly, the unforgability property doesn't hold. For example: in RSA, given a message m with its signature $\sigma = m^d \bmod N$, we can produce the message and signature pair: $(m^2 \bmod N, \sigma^2 \bmod N)$. We now show how to improve the scheme in the random-oracle model where we assume the existence of an oracle access to a random function $h\colon \{0,1\}^* \to \{0,1\}^n$.

Second Attempt - Using an Oracle Access to a Random Function h

1. $\mathrm{Gen}(1^n)$: As previous.
2. $\mathrm{Sign}_{sk}(m)$: Output $\sigma = F_{sk}^{-1}(h(m))$.
3. $\mathrm{Vrfy}_{pk}(m,\sigma)$: output 1 $\iff F_{pk}(\sigma) = h(m)$.

Claim 1 (Consistency)**.** *If* $\sigma = \mathrm{Sign}_{sk}(m)$ *then* $\mathrm{Vrfy}_{pk}(m,\sigma) = 1$*.*

Proof. Holds by the correctness property of the TDP. □

Claim 2 (Existential Unforgability)**.** *The above scheme is existential unforgable under RSA assumption.*

proof of Claim 2. Assume the existence of a PPT algorithm A that breaks the existential unforgability, i.e.

$$\Pr_{(pk,sk)\leftarrow \mathrm{Gen}(1^n)}[\mathrm{A}^{\mathrm{Sign}_{sk}}(1^n, pk) = \left(m^*, F_{sk}^{-1}(h(m^*))\right) \text{ and } \mathrm{Sign}_{sk} \text{ didn't query } m^*] \geq \epsilon,$$

Assume without loss of generality that A always queries the RO (random oracle) on the forged message m^* as well as all signing queries on $m \neq m^*$ before it makes them to the signing oracle

Sign_{sk}. Moreover, assume (without loss of generality) that it makes exactly T queries to the RO, for some polynomial $T = T(n)$, and never makes the same query twice. We now construct an algorithm A^* that breaks the RSA assumption w.p. at least ϵ/T. The algorithm emulates an execution of A where it emulates all the oracle answers to A. This is done as follows:

1. Input Challenge: $y \leftarrow \{0,1\}^n$.
2. Sample a random $i \in [T]$ (trying to hit the location of the forged message).
3. Sample random $x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_T \in \{0,1\}^n$ and compute $y_j = F_{pk}(x_j)$ (for $j \in [T] \setminus i$).
4. Set $y_i = y$.
5. Emulate an execution of A as follows:
   (a) Upon receiving the j'th query to the RO (i.e., $h(m_j)$ for some m_j), answer y_j.
   (b) Upon receiving a query $\text{Sign}_{sk}(m_j)$: If $j = i$, fail and abort. Otherwise, answer x_j.
6. On a successful emulation, output x where (m^*, x) is the output of A in the emulation.

With probability $1/T$, we guess correctly the index i such that $m_i = m^*$. Conditioned on correct guess, the view of A in the emulation is identical to the view of A given a real oracle accesses to a random function h and to Sign_{sk}. By the assumption on A, in this case it should win with probability ϵ, namely it outputs (m^*, x) with $F_{pk}(x) = h(m^*) = y$, which means that we found the preimage of the challenge y. Overall success probability: ϵ/T which implies that $\epsilon \leq \text{negl}(n)$. □

In Practice: People implement signature schemes using a TDP schemes like RSA and replace the random function using a good hash function, like SHA-256.