We introduce digital signatures and present a one-time secure digital signature scheme. We then present collision resistant hashes, and show how they can be used to extend the digital signature scheme to many-time secure.

1 Digital Signatures

A digital signature scheme consists of a three functions:

1. \( \text{Gen}(1^k) \rightarrow (vk, sk) \)
2. \( \text{Sign}(sk, m) \rightarrow \sigma \)
3. \( \text{Verify}(\sigma, m) \rightarrow \{0, 1\} \)

For the correctness of the scheme, we have that
\[
\forall m, vk, sk \text{ where } vk, sk \leftarrow \text{Gen}(1^k) : \text{Verify}(vk, m, \sigma) = 1 = 1
\]

1.1 Game: CMA with existential forger

Here we consider a typical challenger, adversary game where the adversary can choose the message it is forging. We use this game to evaluate a digital signature scheme in Section 1.2.

Challenger Adversary
\[
\begin{align*}
(vk, sk) & \leftarrow \text{Gen}(1^k) \\
\sigma & \leftarrow \text{Sign}(m_1) \\
\sigma & \leftarrow \text{Sign}(m_2) \\
\vdots & \leftarrow \text{Sign}(m_q) \\
\sigma & \leftarrow \text{Sign}(m_q) \\
\sigma^* & \leftarrow \text{Forge signature}
\end{align*}
\]

A wins if \( \text{Ver}(vk, m^*, \sigma^*) = 1 \) and \( \forall i \in [q], m^* \neq m_i \)

1.2 Digital Signature for \( q = 1 \)

We now consider a one-time secure digital signature scheme built using one way functions. \( \text{Gen} \) is now:
1. Sample $x_i^b \leftarrow 0,1^k \forall i \in [k], b \in 0,1$

2. $vk = \begin{bmatrix} f(x_0^0) & \cdots & f(x_k^0) \\ f(x_1^1) & \cdots & f(x_k^1) \end{bmatrix}$

3. $sk = \begin{bmatrix} x_0^0 & \cdots & x_k^0 \\ x_1^1 & \cdots & x_k^1 \end{bmatrix}$

4. $\text{Sign}(sk, m, m \in 0,1^k) : m = m_1 \ldots m_k \rightarrow \sigma = x_1^{m_1}, x_k^{m_k}$

5. $\text{Ver}(vk, m, \sigma) : \text{compute } f(x_m) \text{ for all bits in the message and compare to } vk$

To show that this is secure, we can show that cracking this scheme is equivalent to inverting the one way function. A challenger using the attacker can put $f(x)$ in the verifying key, and the attacker might then need to break the one way function to generate the signature. (50% probability).

We still have two problems: this scheme is only one-time secure, and the scheme only works for small messages. The latter problem arises because the length of the verifying key is equal to the length of the message.

2 Collision resistant hash functions

We can resolve both of these problems by signing a hash of the message, rather than the message itself. If this hash is collision resistant, it has the benefit that even if both messages are chosen, it is still extremely difficult to find a colliding hash, to get the same signature.

$H : 0,1^* \rightarrow 0,1^k$ such that it is hard to find $x, x'$ where $H(x) = H(x')$

2.1 Definition of a family of CRHF

A set of functions $H = h_i : D_i \rightarrow R_i, i \in I$ is a family of collision resistant hash functions if

1. $\text{Gen}(1^k) \rightarrow i \in I$ (the description of i is poly(k))

2. $|R_i| < |D_i|$

3. $(B(i, x) = h_i(x)$ (B is a PPT algorithm)

4. $\forall \text{PPTA} \text{ we have } Pr[i \leftarrow \text{Gen}(1^k), (x, x') \leftarrow A(1^k, i) : h_i(x) = h_i(x') \land x \neq x'] = \text{neg}(k)$

2.2 Discrete log CRHF

Let $G$ be an order $p$ subgroup of $Z_q^*$ where $p$ and $q$ are primes.

Let $g$ be the generator of $G$.

$h \in G$, hard to find $x$ such that $g^x = h$

$(G, g, p) \leftarrow \text{samp}(1^k)$

$Pr[x = A(q, p, g^x)] = \text{neg}(k)$

We can construct a CRHF that halves the size of the input:
1. $(G, g, p, h) \leftarrow \text{Gen}(1^k)$

2. $x \leftarrow \mathbb{Z}_p^*$

3. $h = g^x \cdot H : 0, 1^{2k} \rightarrow 0, 1^k$

4. $H(x, r) = g^x h^r$, $x, r \in 0, 1^k$

To reduce an arbitrary size message into the size that can be signed, one can construct a tree of length-halving hashes. Break the message into $n 2k$ size chunks, and hash each of those. Concatenate them and rechunk them, yielding $n/2 2k$ size chunks. Repeat until the only result is a single $k$ size chunk.

3 Multiple message secure digital signature

Armed with a way to reduce the message to a fixed size, we can now sign messages that contain verifying keys (previously, the size of the verifying key was dependent on the size of the message, so if the message contained a verifying key we were in trouble).

The signing party can generate a new verifying key and signing key with each new message it sends. The new verifying key is concatenated with the message, and is under the signature. In this way, each key is only used once, but along with the message, a new verifying key is “linked” to the sender.