One-time signature scheme

Security Game of one-time signature.

1. $Gen(1^n)$ is run to obtain (pk, sk) pair
2. Adversary A is given the pk. and it asks the signature for a single msg m'. Let σ be the signature. i.e; $\sigma = Sign_{sk}(m')$
3. A outputs (m^*, σ^*). 4. Return 1 if $Verify_{pk}(\sigma^*, m^*) = 1$.

Lamport one-time signature scheme.

$$M = \{0,1\}^3$$

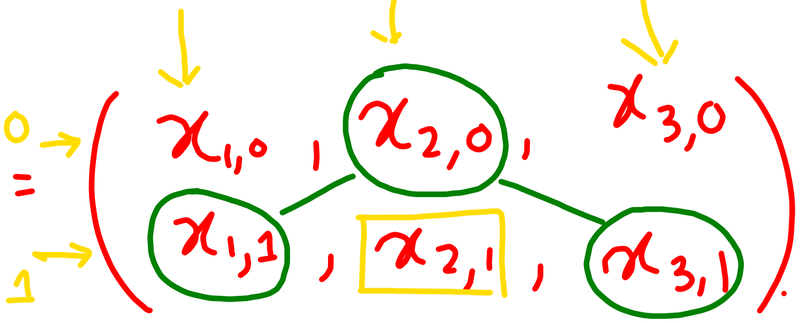
$$\mathrm{KeyGen} :- \mathrm{samples}\begin{pmatrix} x_{1,0}, & x_{2,0}, & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{pmatrix} \leftarrow\!\$\ \{0,1\}^n$$

compute $y_{i,b} = H(x_{i,b})$, where $H : \{0,1\}^n \to \{0,1\}^m$ is a one way fw

$$\text{Set } \langle sk \rangle = \begin{pmatrix} x_{1,0}, & x_{2,0}, & x_{3,0} \\ x_{1,1} & x_{2,1}, & x_{3,1} \end{pmatrix}, \quad \langle pk \rangle = \begin{pmatrix} y_{1,0}, & y_{2,0}, & y_{3,0} \\ y_{1,1} & y_{2,1}, & y_{3,1} \end{pmatrix} := \tilde{Y}$$

Sign:

Let $m = (1\,0\,1)$



signature $\sigma = (x_{1,1}, x_{2,0}, x_{3,1})$

Signer has to produce another msg $m^* \neq m$ and a signature σ^* s.t

$$Verify(pk, m^*, \sigma^*) = 1$$

Verifier receives $m = 101$, $\sigma = (x_{11}, x_{20}, x_{31})$

$$Verify(\tilde{Y}, (m, \sigma)) = H(x_{1,1}) \stackrel{?}{=} y_{1,1} \quad H(x_{3,1}) \stackrel{?}{=} y_{3,1}$$
$$H(x_{2,0}) \stackrel{?}{=} y_{2,0}$$

Adversary for one way fn:

$$P_r\left[A(H(x) \in H^{-1}(H(x))\right] > \frac{1}{p(n)} \quad \text{---(*)}$$

$x \leftarrow \$ \{0,1\}^n$

polynomial exists

If $\exists$ a forging adv A' that breaks the one-time signature scheme, then $\exists$ A that breaks the one wayness of H as defined in (*)

Stateful signature scheme.

KeyGen(1^n) :- it generates a public key, secret key $\langle pk, sk \rangle$ pair and it also generates a state

outputs (pk, sk, s_0)

Sign: it takes the sk, msg m, and a state s_{i-1} to produce a signature σ and updates the state to s_i and it stores it.

Verify: it takes the public key pk, the msg m and a signature σ and it outputs 0/1.

l - known upper bound on the number of signatures produced.

KeyGen (1^n): - it generates $pk = \langle pk_1, \ldots, pk_l \rangle$ and $sk = \langle sk_1 \ldots, sk_l \rangle$ by invoking the keyGen alg of a one-time signature scheme for l-times. and it initializes the state to 1.

Sign: on i^{th} message m_i, $\sigma_i \leftarrow \text{Sign-OTS}_{sk_i}(m_i)$, and it updates the state to $(i+1)$. and it outputs (σ_i, i)

Verify: on i/p $(\sigma, j, \hat{m})$ it runs $\text{Verify-OTS}_{pk_j}(\sigma, \hat{m})$. If o/p 1 iff $\text{Verify-OTS}_{pk_j}(\sigma, \hat{m}) = 1$

Chain-based signature scheme.

KeyGen (1^n): it generates $\langle pk, sk \rangle$ pair

$\langle pk_1, sk_1 \rangle$

Sign: it signs on a msg m_1 ← first time.

it generates a random string (pk_2, sk_2) pair

and it generates $\sigma_1 = Sign_{sk_1}(m_1 \| pk_2)$

it stores the state = $(m_1, \sigma_1, pk_2, sk_2)$.

Signing on msg m_2

- it generates $\langle pk_3, sk_3 \rangle$ pair
- computes $\sigma_2 \leftarrow Sign_{sk_2}(m_2 || pk_3)$
- stores the tuple $(m_2, \sigma_2, pk_3, sk_3)$

$\{(m_1, \sigma_1, pk_2, sk_2),$
$(m_2, \sigma_2, pk_3, sk_3)\}.$
$(m_3, \sigma_3, pk_4, sk_4)\}.$

Signing on msg m_3.

$\sigma_3 \leftarrow Sign_{sk_3}(m_3 || pk_4) \rightarrow$ O/P $(m_3 || pk_4, \sigma_3, \{(m_1, \sigma_1, pk_2, sk_2),$
$(m_2, \sigma_2, pk_3, sk_3)\}.)$

Stores $(m_3, \sigma_3, pk_4, sk_4)$

Verifier has access to pk_1

Verifies pk_1

$m_3, \sigma_3, \{(m_1, \sigma_1, pk_2, sk_2), (m_2, \sigma_2, pk_3, sk_3)\}$

$\sigma_1 \leftarrow Sign_{sk_1}(m_1 \| pk_2)$

(i) $Verify_{pk_1}(\sigma_1, m_1 \| pk_2) = 1 \quad \wedge$

(ii) $Verify_{pk_2}(\sigma_2, m_2 \| pk_3) = 1 \quad \wedge$

(iii) $Verify_{pk_3}(\sigma_3, m_3 \| pk_4) = 1$

m -

$$sk = \begin{pmatrix} x_{10} & x_{20} & \cdots & x_{l0} \\ x_{11} & x_{21} & \cdots & x_{l1} \end{pmatrix}$$

$$\sigma = (x_{ib})_{i=1}^{l} \qquad x_{ib} = \begin{cases} x_{i0} & \text{if } m_i = 0 \\ x_{i1} & \text{if } m_i = 1 \end{cases}$$

$$\text{Verify}(\tilde{Y}, (m, \sigma)) = H(x_{ib}) \stackrel{?}{=} y_{ib} \quad \begin{matrix} i = 1, \cdots, l \\ b = 0, 1 \end{matrix}$$