Hash-then-Sign (RSA-FDH)

$pk = (N, e)$, $sk = (N, d)$ $\quad H: \{0,1\}^* \to \mathbb{Z}_N^*$

$$Sign_{SK}(m) = (H(m))^d \bmod N.$$

$$Verify_{PK}(m, \sigma) = \sigma^e \stackrel{?}{=} H(m) \bmod N$$

Thm: If the RSA problem is hard relative to GenRSA and H is modelled as a random oracle, then RSA-FDH is secure.

Assumption: When adversary makes query to the signing oracle with m, it must be the case that 'm' must have been queried to the random oracle before.

$Sig\text{-}forge_{A,\Pi}(n)$.

- Run $Gen_{RSA}(1^n)$ and obtain (N, e, d).
  Then a random $H: \{0,1\}^* \to \mathbb{Z}_N^*$ is chosen.
- The adversary A is given $pk = (N, e)$ and may query to the random oracle $H(\cdot)$, or to the signing oracle $Sign_{SK}(\cdot)$ with i/p message m, and it obtains

$$\sigma = (H(m))^d \bmod N.$$

$Sig\text{-}forge_{A,\Pi}(n)$.

- Run $Gen_{RSA}(1^n)$ and obtain (N, e, d).
  Then a random $H: \{0,1\}^* \to \mathbb{Z}_N^*$ is chosen.
- The adversary A is given $pk = (N, e)$ and may query to the random oracle $H(\cdot)$, or to the signing oracle $Sign_{SK}(\cdot)$ with i/p message m, and it obtains.

$$\sigma = (H(m))^d \bmod N.$$

- A submits (m^*, σ^*) pair such that m^* does not belong to the set of signing queried messages

The o/p of the experiment is 1 iff

$(\sigma^*)^e = H(m^*) \bmod N$.

- A submits (m^*, σ^*) pair such that m^* does not belong to the set of signing queried messages

The o/p of the experiment is 1 iff

$(\sigma^*)^e = H(m^*) \bmod N$.

Ch

A

$(N, e, d) \leftarrow Gen\text{-}RSA(1^n)$

$\xleftarrow{n}$

Choose $H: \{0,1\}^* \to \mathbb{Z}_N^*$

Compute $H(m)$

Check if $H(m)$ is already in the list L.

if exists, then computes

$\sigma = H(m)^d \bmod N$

$\xleftarrow{m}$

$\xrightarrow{\sigma}$

Sig-forge$'_{A,\pi}(n)$.

Assume A makes $q := q(n)$ many distinct queries to H.

1. Choose an uniform $j \in \{1, \ldots q\}$.

2. Run $GenRSA(1^n)$ and obtain (N, e, d).
   Choose an uniform $H: \{0,1\}^* \to \mathbb{Z}_N^*$.

3. Run A with (N, e); which can query to H or the signing oracle with m and receives $\sigma = H(m)^d \bmod N$.

4. A outputs (m^*, σ^*), where it had not previously requested for a signature on m. Let i be such that $m^* = m_i$. The o/p of the experiment is 1 if (i) $\sigma^{*e} = H(m^*) \bmod N$ and (ii) $i = j$

Sig-forge$'_{A,\pi}(n)$.

Assume A makes $q := q(n)$ many distinct queries to H.

1. Choose an uniform $j \in \{1, \ldots q\}$.

2. Run $GenRSA(1^n)$ and obtain (N, e, d).
   Choose an uniform $H: \{0,1\}^* \to \mathbb{Z}_N^*$.

3. Run A with (N, e); which can query to H or the signing oracle with m and receives $\sigma = H(m)^d \bmod N$.

4. A outputs (m^*, σ^*), where it had not previously requested for a signature on m. Let i be such that $m^* = m_i$. The o/p of the experiment is 1 if (i) $\sigma^{*e} = H(m^*) \bmod N$ and (ii) $i = j$

$$\Pr\left[\text{Sig-forge}'_{A,\Pi}(n) = 1\right].$$

$$= \Pr\left[i = j \wedge \text{Sig-forge}_{A,\Pi}(n) = 1\right]$$

$$= \frac{1}{q} \cdot \Pr\left[\text{Sig-forge}_{A,\Pi}(n) = 1\right].$$

$\underline{\text{Sig-forge}''_{A,\Pi}(n)}$. Same as $\text{Sig-forge}'_{A,\Pi}(n)$, but if A ever request a signature on message m_j, then the challenger aborts.

$$P[\text{Sig-forge}''_{A,\pi}(n) = 1] = P[\text{Sig-forge}'_{A,\pi}(n) = 1]$$

$$f : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$$
$$f(\sigma) = \sigma^e$$

$$= \frac{1}{q} \cdot P[\text{Sig-forge}_{A,\pi}(n) = 1]$$

Let A be an adversary that wins $\text{Sig-forge}''_{\pi}(n)$ game.
Then we construct an adversary A' that solves RSA problem.

m_1 $\quad y_1 = H(m_1)$

$y_1 \leftarrow \$ \mathbb{Z}_N^*$

	Sig	H
m_1	σ_1	σ_1^e

$$\sigma = H(m)^d \bmod N$$
$$\Rightarrow \sigma^e = H(m) \bmod N$$

Algorithm A'

I/p: (N, e, y).

1. choose $j \leftarrow\$ \{1, \ldots, q\}$
2. Run A on the $pk = (N, e)$. Store triplets $(\cdot, \cdot, \cdot)$ in a table, which is initially empty. An entry (m_i, σ_i, y_i) indicates that A' has set $H(m_i) = y_i$ and $\sigma_i^e = y_i \bmod N$.
3. when A makes it's i^{th} random oracle query $H(m_i)$, answer it as follows:
   (i) if $i \neq j$, $\sigma_i \leftarrow\$ \mathbb{Z}_N^*$ and compute $y_i = \sigma_i^e \bmod N$ and store (m_i, σ_i, y_i) in the table

Algorithm A'

I/p: (N, e, y).

1. choose $j \leftarrow\$ \{1, \ldots, q\}$
2. Run A on the $pk = (N, e)$. Store triplets $(\cdot, \cdot, \cdot)$ in a table, which is initially empty. An entry (m_i, σ_i, y_i) indicates that A' has set $H(m_i) = y_i$ and $\sigma_i^e = y_i \bmod N$.
3. when A makes it's i^{th} random oracle query $H(m_i)$, answer it as follows:
   (i) if $i \neq j$, $\sigma_i \leftarrow\$ \mathbb{Z}_N^*$ and compute $y_i = \sigma_i^e \bmod N$ and store (m_i, σ_i, y_i) in the table and return y_i as the response.

(ii) if $i=j$ then return y.

When A requests a signature on message m, let i be such that $m = m_i$ and answer the query as follows:

- if $i=j$ then A' aborts
- if $i \neq j$, then there is a entry (m_i, σ_i, y_i) in the table. Return σ_i as the response.

At the end of the execution of A, it outputs $(\overset{*}{m}, \overset{*}{\sigma})$. If $\overset{*}{m} = m_j$ and $\overset{*}{\sigma}^e = y \bmod N$, then return σ^*.

$$\sigma^{*e} = H(\overset{*}{m}) \bmod N$$

$$\Pr[\mathrm{RSA\text{-}Inv}_{A',\mathrm{GenRSA}}(n) = 1].$$

$$\geq \Pr[\mathrm{Sig\text{-}forge}''_{A,\Pi}(n) = 1].$$

$$= \frac{1}{q} \cdot \Pr[\mathrm{Sig\text{-}forge}_{A,\Pi}(n) = 1]$$

$$\Rightarrow \Pr[\mathrm{Sig\text{-}forge}_{A,\Pi}(n) = 1] \leq q \cdot \Pr[\mathrm{RSA\text{-}Inv}_{A',\mathrm{GenRSA}}(n) = 1].$$

Identification Scheme.

Prover (sk) — Verifier (pk).

$(I, st) \leftarrow P_1(sk)$.

I →

← $r \leftarrow\$ \Omega_{pk}$.

$s := P_2(sk, st, r)$.

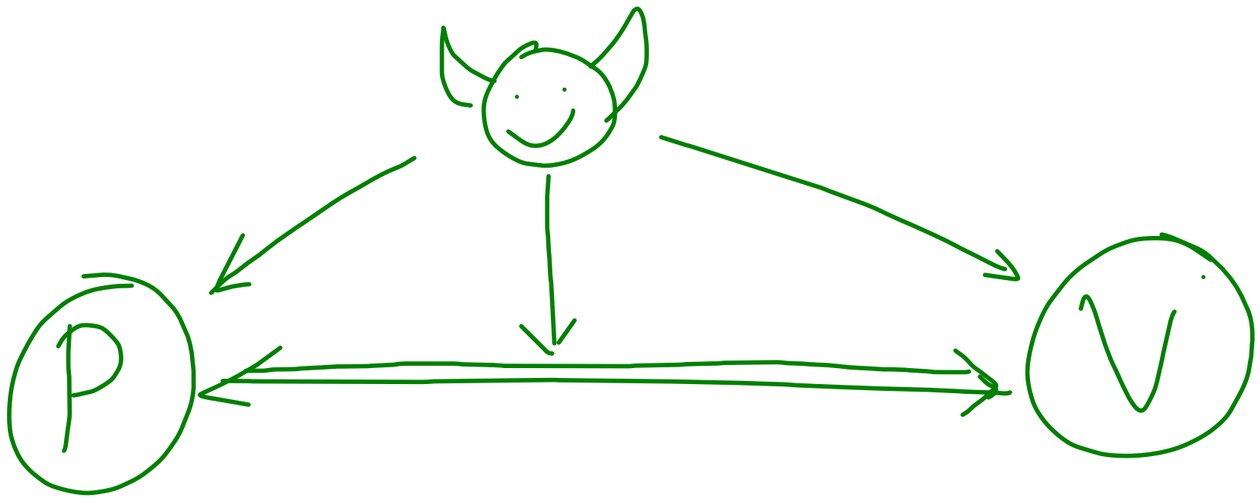
s. →

$V(pk, r, s) \stackrel{?}{=} I$.

(P_1, P_2, V) are three poly-time algorithms.

Security of Identification Scheme.

1. $Gen(1^n)$ is run to obtain (pk, sk).
2. Adversary A is given pk and access to an oracle $Trans_{sk}$.
3. At any point during the experiment, A outputs a message I. A uniform challenge $r \in \Omega_{pk}$ is chosen and given to A, who responds with some s. (A can query to $Trans_{sk}$ after receiving r).
4. The experiment will o/p 1 if $V(pk, r, s) = I$.


P
V