Digital Signature

Message Authentication Codes.

- Integrity of the msg.
- Authenticity.

$t \leftarrow MAC(k, m)$.

Digital Signature is the public key counterpart of MAC.

$S \leftarrow K \rightarrow R$

Defn: A digital signature scheme Π consists of three polynomial time algorithms.

- KeyGen(1^n) :- it takes 1^n as i/p and outputs a (pk, sk) pair.
- Sign(sk, m) :- it takes the secret key and the message and it produces a signature σ.
- Verify(pk, m, σ) :- it takes the public key, the message and a signature, and it outputs either 0 or 1.

Correctness:

$\forall (pk, sk), \forall m$

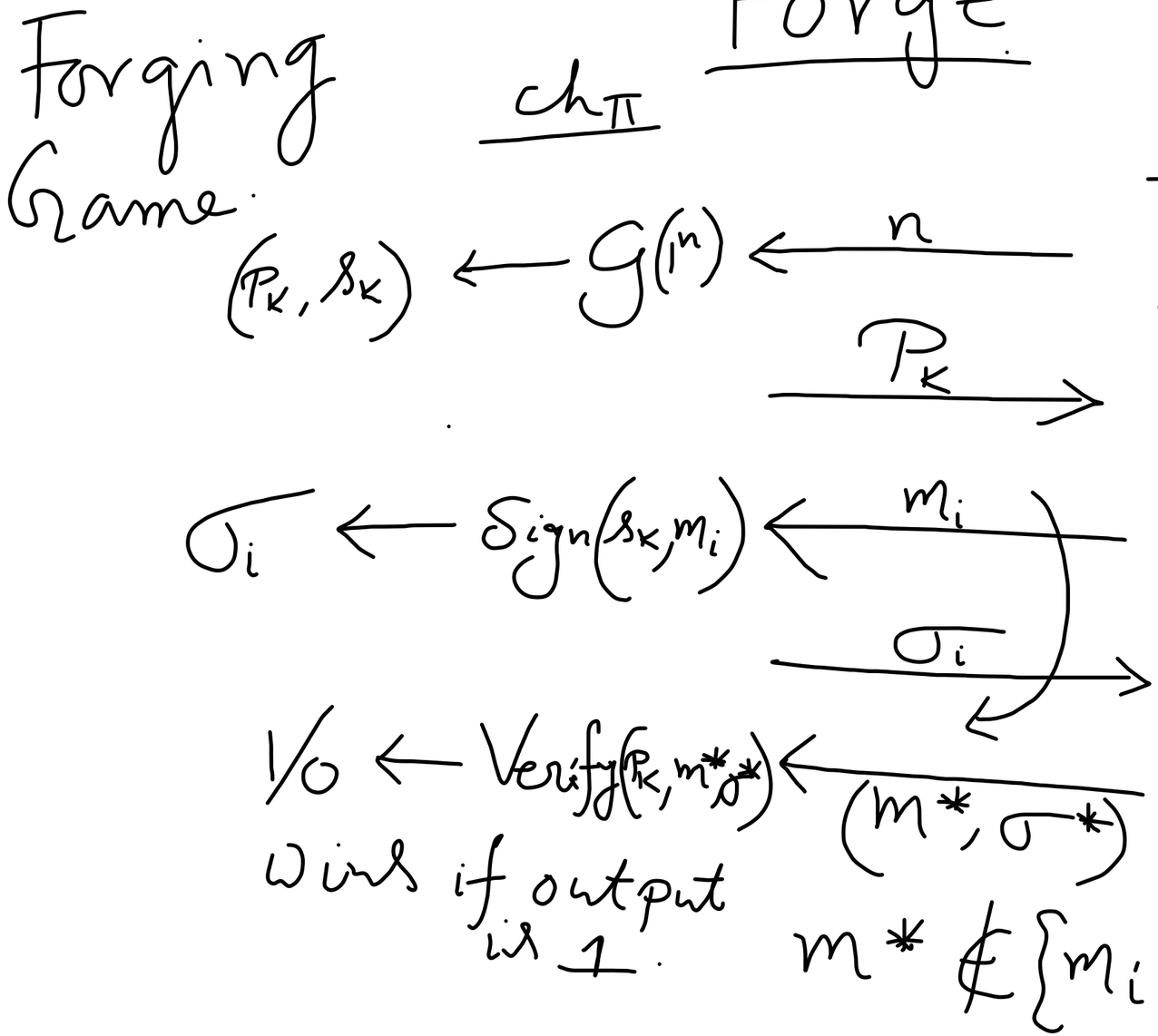
$$Verify(pk, m, Sign(sk, m)) = 1.$$

Signature σ is a valid signature of msg. m.

Security Game of Digital Signature Scheme.

Forge

Forging Game



Defn: We call a digital signature scheme Π is secure (existentially unforgeable) if for every PPT algorithm D, $\exists$ a negligible fn negl(.) such that

$$Pr[Forge_\Pi(D)=1] \leq negl(n)$$

RSA-Signature scheme.

RSA-Signature scheme

KeyGen (1^n):

1. Sample two n-bit odd primes p, q.
2. Compute $N = p \cdot q$ and $\phi(N) = (p-1)(q-1)$
3. Choose $e \in \mathbb{Z}_N^*$ such that $\gcd(e, \phi(N)) = 1$
4. Compute d such that $ed \equiv 1 \bmod \phi(N)$.
5. output $pk = (N, e)$, $sk = (N, d)$.

Sign (sk, m).

Computes $\sigma = m^d \bmod N$; where $m \in \mathbb{Z}_N^*$.

Verify (pk, m, σ).

Checks whether $\sigma^e \equiv m \bmod N$.

Verify the correctness.

$$\text{Verify}\big((N,e), m, \text{Sign}((N,d), m)\big) = \text{Verify}\big((N,e), m, m^d \bmod N\big)$$
$$= (m^d)^e \bmod N \stackrel{?}{=} m.$$

Hash-then-Signature scheme.

$\Pi' = (\text{KeyGen}, \text{Sign}', \text{Verify}')$ - fixed signature scheme

Π.KeyGen(1^n): on i/p 1^n, it outputs (pk, sk) and a hash fn $H: \{0,1\}^* \to G$.

Π.Sign(sk, m) : → computes $\sigma = \text{Sign}'(sk, H(m))$.

Signing alg. of a fixed length signature scheme.

Π.Verify(pk, m, σ) - it output 1 if Verify$'(pk, H(m), \sigma) = 1$

Is RSA-signature scheme secure?

Is RSA-signature scheme secure?

- (i) No-message attack
- (ii) Attack by exploiting multiplicative rel^{n}

↓

Access to the signing oracle.

$S = \{m_1, \cdots, m_q\}$ - $(2^q - (q+1))$.

RSA-FDH (RSA- Full domain hash).

KeyGen(1^n):- on i/p n, it samples two n-bit odd primes (p, q).

- Compute $N = pq$ and $\phi(N) = (p-1)(q-1)$
- Choose $e \in \mathbb{Z}_N^*$ such that $\gcd(e, \phi(N)) = 1$.
- Compute $d = e^{-1} \bmod \phi(N)$.
- Sample a hash fn $H: \{0,1\}^* \to \mathbb{Z}_N^*$
- Set the public key $pk = (N, e)$ and $sk = (N, d)$. and a hash fn

Sign(sk, M)

- computes $s = H(M)$.
- computes $\sigma = s^d \bmod N$.
- returns σ.

$\sigma = Sign'(sk, H(m))$.

We want

(a) No message attack works for RSA-FDH.

- Hash fn should have the one-wayness.

(b) Multiplicative rel^n should not hold.

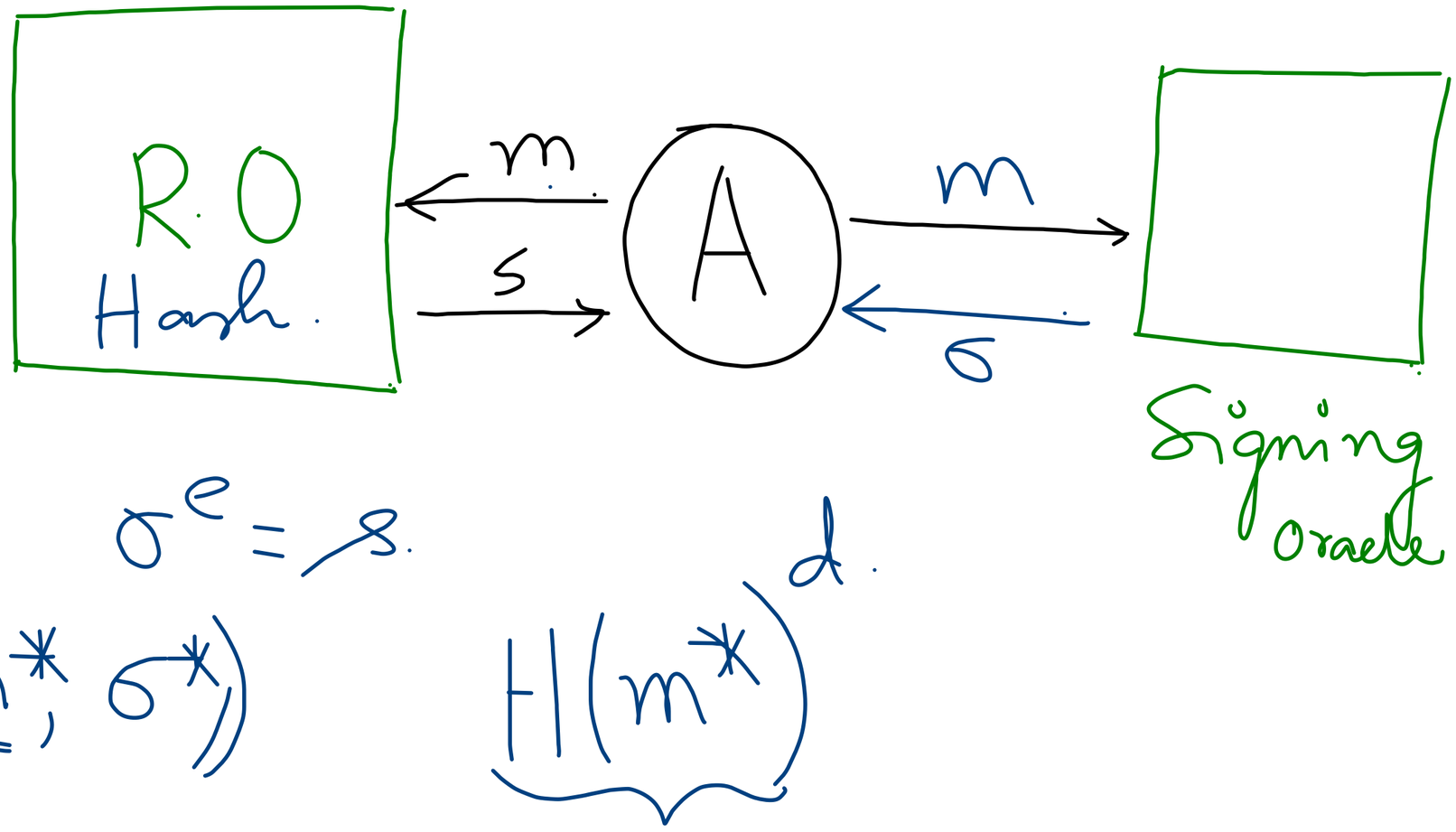
Verify (pk, M, σ)

- computes $s = H(M)$.
- returns 1 if $\sigma^e = s \bmod N$.

$\sigma_1^e = H(m_1)$, $\sigma_2^e = H(m_2)$. $\sigma_1^e \cdot \sigma_2^e = H(m_1) \cdot H(m_2)$

$m : H(m) = H(m_1) \cdot H(m_2)$ (X)

RSA-FDH.



$\sigma^e = s.$

(m^*, σ^*) $\qquad$ $H(m^*)^d.$