# Digital Signature

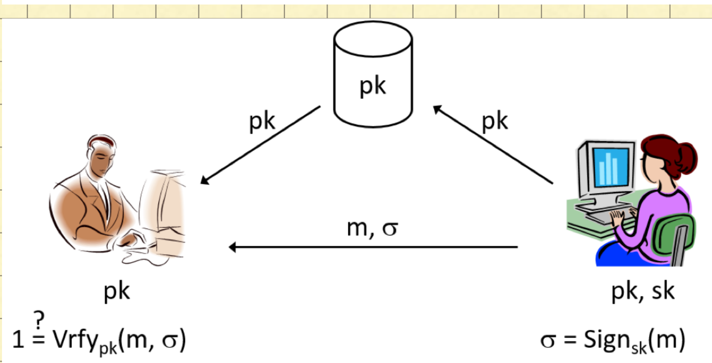
- Provide *integrity* in the public-key setting
- Analogous to *message authentication codes*, but some key differences

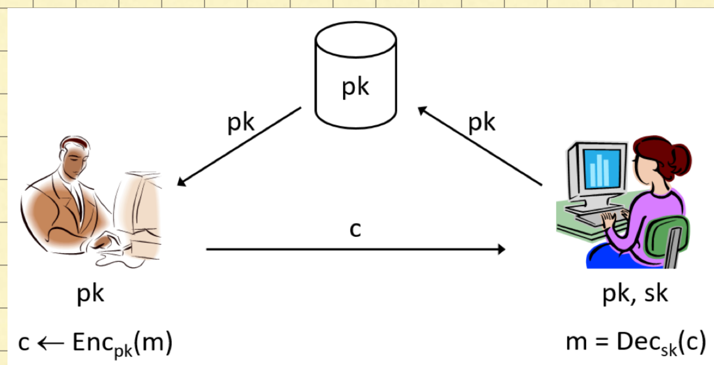
| | Private Key | Public Key |
|---|---|---|
| **Secrecy** | private key encryption | public key encryption |
| **Integrity** | MAC | ?? |

- A *signature scheme* is defined by three PPT algorithms (*Gen*, *Sign*, *Vrfy*):
  - *Gen*: takes as input $1^n$; outputs $pk, sk$
  - *Sign*: takes as input a private key $sk$ and a message $m \in \{0,1\}^*$; outputs *signature* $\sigma$: $\sigma \leftarrow Sign_{sk}(m)$
  - *Vrfy*: takes public key $pk$, message $m$, and signature $\sigma$ as input; outputs 1 or 0

  For all $m$ and all $pk$, $sk$ output by *Gen*,
  $$Vrfy_{pk}(m, Sign_{sk}(m)) = 1$$

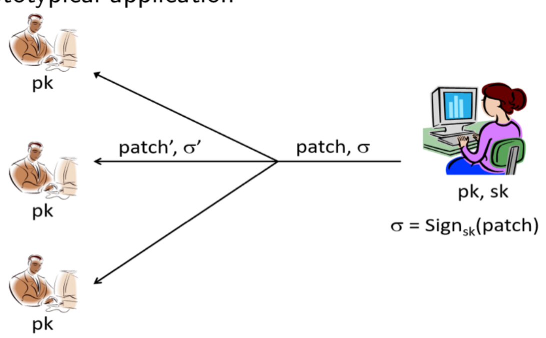
私钥做签名
公钥做验证



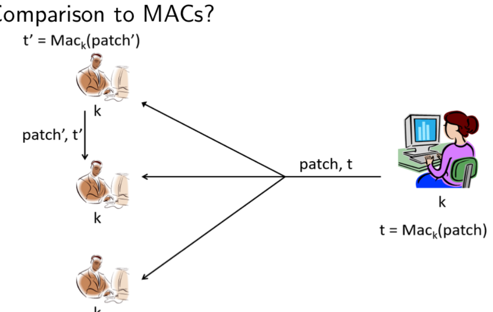

# Security (informal)

- Even after observing signatures on multiple messages, an attacker should be unable to *forge* a valid signature on a new message
- Prototypical application

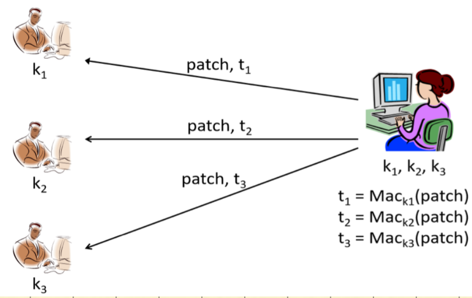



可以观察到多个message的签名，
但不能对新的message伪造签名.

- Comparison to MACs?




- Comparison to MACs?



# Comparison to MACs

- *Public verifiability*
  - "Anyone" can verify a signature
  - Only a holder of the key can verify a MAC tag

⇒ *Transferability*
  - Can forward a signature to someone else

⇒ *Non-repudiation*

所有人都可以验证签名
但是MAC tag不能，因为MAC的密钥是事先商定且不公开的.

不可否认性：一旦签名验证通过，就无法否认信息有效性.

# Non-repudiation

- Signer cannot (easily) deny issuing a signature
  - Crucial for legal applications
  - Judge can verify signature using public copy of $pk$
- MACs cannot provide this functionality!
  - Without access to the key, no way to verify a tag
  - Even if receiver leaks key to judge, how can the judge verify that the key is correct?
    - Even if key is correct, receiver could have generated the tag also!

# Security

- Threat model
  - "*Adaptive chosen-message attack*"
  - Assume the attacker can induce the sender to sign *messages of the attacker's choice*
- Security goal
  - "*Existential unforgeability*"
  - Attacker should be unable to forge valid signature on any message not signed by the sender
- Attacker gets the public key

按攻击者的选择产生签名.

# Formal definition

- **Definition 14.1** Fix $A, \Pi$. Define randomized experiment $Forge_{A,\Pi}(n)$:
  1. $pk, sk \leftarrow Gen(1^n)$
  2. A is given $pk$, and interacts with *oracle* $Sign_{sk}(n)$; let $M$ be the set of messages sent to this oracle
  3. A outputs $(m, \sigma)$
  4. A *succeeds*, and the experiment evaluates to 1, if $Vrfy_{pk}(m, \sigma) = 1$ and $m \notin M$

  $\Pi$ is *secure* if for all PPT attackers $A$, there is a negligible function $\epsilon$ such that

$$\Pr[Forge_{A,\Pi}(n) = 1] \leq \epsilon(n)$$

# Replay attacks

先对明文hash之后再签名

- Replay attacks need to be addressed just as in the symmetric-key setting
  - The *hash-and-sign paradigm*
- Given
  - A signature scheme $\Pi = (Gen, Sign, Vrfy)$ for "short" messages of length $n$
  - Hash function $H : \{0,1\}^* \to \{0,1\}^n$
- Construct a signature scheme $\Pi' = (Gen, Sign', Vrfy')$ for arbitrary-length messages:
  - $Sign'_{sk}(m) = Sign_{sk}(H(m))$
  - $Vrfy'_{pk}(m, \sigma) = Vrfy_{pk}(H(m), \sigma)$

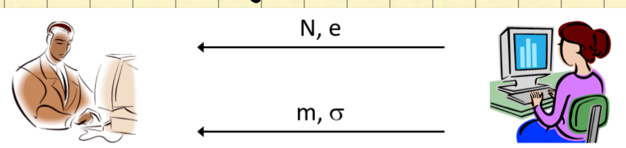
# Hash-and-sign paradigm

- **Theorem 14.2** If $\Pi$ is *secure* and $H$ is *collision-resistant*, then $\Pi'$ is *secure*.

**Proof.** Say the sender authenticates $m_1, m_2, \ldots$
- Let $h_i = H(m_i)$

Attacker outputs forgery $(m, \sigma)$, $m \neq m_i$ for all $i$
Two cases:
- $H(m) = h_i$ for some $i$
  - Collision in $H$!
- $H(m) \neq h_i$ for all $i$
  - Forgery in the underlying signature scheme!

# Plain RSA signature




$m \stackrel{?}{=} [\sigma^e \bmod N]$

$(N, e, d) \leftarrow RSAGen(1^n)$
pk = (N, e)
sk = d

$\sigma = [m^d \bmod N]$

**Key generation**: choose two random $p, q$ and compute $N = p \cdot q$. Run $GenRSA(1^n)$. The secret key is $(N, e)$. The public key is $(N, d)$.

**Signing**: To sign a message $m$, output $\sigma = m^d \pmod{n}$.
**Verification**: To verify that $\sigma$ is a valid signature for $m$, check whether $\sigma^e = m \pmod{n}$.

# Security

- Intuiation
  - Signature of $m$ is the $e^{th}$ root of $m$ – supposedly hard to compute
- Attack1: Can sign *specific* messages
  - E.g., easy to compute the $e^{th}$ root of $m = 1$, or the cube root of $m = 8$
- Attack2: Can sign "*random*" messages
  - Choose arbitrary $\sigma$; set $m = \sigma^e \pmod{N}$
- Attack3: Can combine two signatures to obtain a third
  - Say $\sigma_1, \sigma_2$ are valid signatures on $m_1, m_2$ w.r.t. public key $N, e$
  - Then $\sigma' = \sigma_1 \cdot \sigma_2 \bmod N$ is a valid signature on the message $m' = m_1 \cdot m_2 \bmod N$

# RSA-FDH (Full Domain Hash) Signature Scheme

- Main idea: apply a "*cryptographic transformation*" to messages before signing

- **Construction 14.3**: Construct a signature scheme as follows:
  - *Gen*: on input $1^n$, run $GenRSA(1^n)$ to compute $(N, e, d)$. The public key is $(N, e)$, and the private key is $d$. As part of key generation, a function $H: \{0,1\}^* \to \mathbb{Z}_N^*$ is specified.
  - $Sign_{sk}(m)$: on input a private key $(N, d)$ and a message $m \in \{0,1\}^*$, compute $Sign_{sk}(m) = \sigma = H(m)^d \bmod N$
  - $Vrfy_{pk}(m, \sigma)$: On input a public key $(N, e)$, a message $m$, and a signature $\sigma$, output 1 if and only if $\sigma^e = H(m) \bmod N$

哈希函数加入了随机性.

## Security

- Look at the three previous attacks:
  - Not easy to compute the $e^{th}$ root of $H(1)$ ...
  - Choose $\sigma$, but how do you find an $m$ s.t. $H(m) = \sigma^e \bmod N$?
    - Computing *inverses* of $H$ should be hard
  - $H(m_1) \cdot H(m_2) = \sigma_1^e \cdot \sigma_2^e = (\sigma_1 \cdot \sigma_2)^e \neq H(m_1 \cdot m_2)$

- **Theorem 14.4** If the *RSA assumption* holds, and $H$ is modeled as a *random oracle* (mapping onto $\mathbb{Z}_N^*$), then RSA-FDH is secure.

## RSA-FDH in practice

- In practice, $H$ is instantiated with a modified cryptographic hash function
  - Must ensure that the range of $H$ is large enough

- The RSA PKCS #1 v2.1 standard includes a signature scheme inspired by RSA-FDH
  - Essentially a randomized variant of RSA-FDH

- DSS: NIST standard for digital signatures
  - DSA, based on *discrete-logarithm problem* in subgroup of $\mathbb{Z}_p^*$
  - ECDSA, based on elliptic-curve groups

## Plain Rabin signature

- **Key generation**: choose two random $p, q$ with $p, q \equiv 3 \pmod 4$, as secret keys. The public key is $n = p \cdot q$.

  **Signing**: To sign a message $m$, output $\sigma = \sqrt{m} \pmod n$ (fix some choice for one of the four possible roots).

  **Verification**: To verify that $\sigma$ is a valid signature for $m$, check whether $\sigma^2 = m \pmod n$.
- **Note**: Assuming the *factoring problem* is hard, if $m$ is chosen at random, then it should be hard to forge a signature for $m$.
- However, this scheme is *insecure* against *chosen-message attack*.
  - Choose an $x \in \mathbb{Z}_n^*$ at random, and let $m = x^2 \pmod n$
  - Given $\sigma = \sqrt{m} \pmod n$ there is probablity $1/2$ that $\sigma \neq \pm x \pmod n$ in which case $\gcd(\sigma - x, n)$ will yield a nontrivial factor of $n$