

# Digital Signature

Yu Zhang

Harbin Institute of Technology

Cryptography, Autumn, 2021

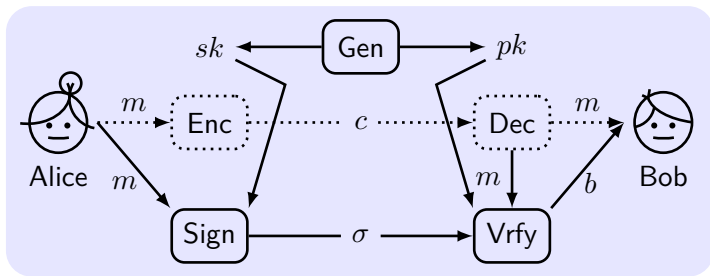
- 1** Definitions of Digital Signatures
- 2** RSA Signatures
- 3** Digital Signature from The Discrete-Log Problem
- 4** One-Time Signature and Signature from Hash
- 5** Certificates and Public-Key Infrastructures

- 1 Definitions of Digital Signatures**
- 2 RSA Signatures
- 3 Digital Signature from The Discrete-Log Problem
- 4 One-Time Signature and Signature from Hash
- 5 Certificates and Public-Key Infrastructures

# Digital Signatures – An Overview

- **Digital signature scheme** is a mathematical scheme for demonstrating the authenticity/integrity of a digital message
- Allow a **signer**  $S$  to “**sign**” a message with its own  $sk$ , anyone who knows  $S$ 's  $pk$  can **verify** the authenticity/integrity
- (Comparing to MAC) digital signature is:
  - publicly verifiable
  - transferable
  - non-repudiation
  - but slow
- Q: What are the differences between digital signatures and handwritten signatures?
- Digital signature is NOT the “inverse” of public-key encryption
- Signatures are used to convey *trust* from a public key to the data which is signed

# The Syntax of Digital Signature Scheme



- signature  $\sigma$ , a bit  $b$  means valid if  $b = 1$ ; invalid if  $b = 0$ .
- **Key-generation** algorithm  $(pk, sk) \leftarrow \text{Gen}(1^n), |pk|, |sk| \geq n$ .
- **Signing** algorithm  $\sigma \leftarrow \text{Sign}_{sk}(m)$ .
- **Verification** algorithm  $b := \text{Vrfy}_{pk}(m, \sigma)$ .
- **Basic correctness requirement:**  $\text{Vrfy}_{pk}(m, \text{Sign}_{sk}(m)) = 1$ .

# Defining Signature Security

The signature experiment  $\text{Sigforge}_{\mathcal{A}, \Pi}(n)$ :

- 1  $(pk, sk) \leftarrow \text{Gen}(1^n)$ .
- 2  $\mathcal{A}$  is given input  $1^n$  and oracle access to  $\text{Sign}_{sk}(\cdot)$ , and outputs  $(m, \sigma)$ .  $\mathcal{Q}$  is the set of queries to its oracle.
- 3  $\text{Sigforge}_{\mathcal{A}, \Pi}(n) = 1 \iff \text{Vrfy}_{pk}(m, \sigma) = 1 \wedge m \notin \mathcal{Q}$ .

## Definition 1

A signature scheme  $\Pi$  is **existentially unforgeable under an adaptive CMA** if  $\forall$  PPT  $\mathcal{A}$ ,  $\exists$   $\text{negl}$  such that:

$$\Pr[\text{Sigforge}_{\mathcal{A}, \Pi}(n) = 1] \leq \text{negl}(n).$$

**Q: What's the difference on the ability of adversary between MAC and digital signature? What if an adversary is not limited to PPT?**

- 1 Definitions of Digital Signatures
- 2 RSA Signatures**
- 3 Digital Signature from The Discrete-Log Problem
- 4 One-Time Signature and Signature from Hash
- 5 Certificates and Public-Key Infrastructures

# Insecurity of “Textbook RSA”

## Construction 2

- Gen: on input  $1^n$  run  $\text{GenRSA}(1^n)$  to obtain  $N, e, d$ .  
 $pk = \langle N, e \rangle$  and  $sk = \langle N, d \rangle$ .
- Sign: on input  $sk$  and  $m \in \mathbb{Z}_N^*$ ,  $\sigma := [m^d \bmod N]$ .
- Vrfy: on input  $pk$  and  $m \in \mathbb{Z}_N^*$ ,  $m \stackrel{?}{=} [\sigma^e \bmod N]$ .
- **No-message attack:** choose an arbitrary  $\sigma \in \mathbb{Z}_N^*$  and compute  $m := [\sigma^e \bmod N]$ . Output the forgery  $(m, \sigma)$ .

$$pk = \langle 15, 3 \rangle, \sigma = 2, m = ? \quad m^d = ?$$

- **Arbitrary message attack:** To forge a signature on  $m$ , choose a random  $m_1$ , set  $m_2 := [m/m_1 \bmod N]$ , obtain signatures  $\sigma_1, \sigma_2$  on  $m_1, m_2$ .

Q:  $\sigma := [\text{____} \bmod N]$  is a valid signature on  $m$ .



# Hashed RSA Signatures

**Idea:** Use hash function to break the strong algebraic relationship between the message and the signature.

**RSA-FDH Signature Scheme:** Random Oracle as a **Full Domain Hash (FDH)** whose image size = the RSA modulus  $N - 1$ .

- Gen: a hash function  $H : \{0, 1\}^* \rightarrow \mathbb{Z}_N^*$  is part of public key.
- Sign:  $\sigma := [H(m)^d \bmod N]$ .
- Vrfy:  $\sigma^e \stackrel{?}{=} H(m) \bmod N$ .
- **No-message attack:** Cannot invert  $H(m) := \sigma^e \bmod N$
- **Arbitrary message attack:**  $H(m/m_1)$  has no relationship with  $H(m)$  and  $H(m_1)$

## Insecurity

There is NO known function  $H$  for which hashed RSA signatures are secure.

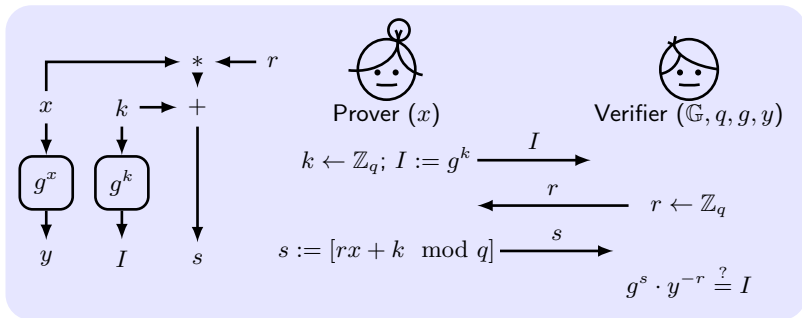
- 1 Definitions of Digital Signatures
- 2 RSA Signatures
- 3 Digital Signature from The Discrete-Log Problem**
- 4 One-Time Signature and Signature from Hash
- 5 Certificates and Public-Key Infrastructures

# Overview of Schnorr Signature

- **Schnorr signature** (1988) is a typical instance showing the relationships among signature, identification and zero-knowledge proof.
- **Construction:** non-interactive protocol of Schnorr identification protocol, which is an interactive zero-knowledge proof protocol for DLP solution.
- **Security:** proved by applying the Fiat–Shamir transformation to Schnorr identification protocol in ROM and under the assumption of DLP hardness.
- **Applications:** multisignature, threshold signature and blind signature, which are heavily used in cryptocurrency.

# Schnorr Identification Scheme

The prover **publicly** proves that she is the one who knows the solution  $x$  of a DLP  $y$  in a 3-round  $\Sigma$ -protocol.



## Why not a simpler identification protocol?

First, the verifier generates  $r$  and sends  $g^r$ ; then the prover replies with  $s = g^{rx}$ ; last, the verifier checks  $y^r \stackrel{?}{=} s$ .

# Proof of Schnorr Identification Scheme

## Theorem 3

*If the discrete-log problem is hard, then the Schnorr identification scheme is secure.*

**Idea:** If the attacker can let  $g^s \cdot y^{-r} = I$ , then the attacker can compute  $x$ .

## Proof.

Reduce  $\mathcal{A}'$  inverting  $y$  to  $\mathcal{A}$  attacking the Schnorr scheme:

- 1  $\mathcal{A}'$  as a verifier, answering all queries, runs  $\mathcal{A}$  as a prover.
- 2 When  $\mathcal{A}$  outputs  $I$ ,  $\mathcal{A}'$  chooses  $r_1 \in \mathbb{Z}_q$  and send to  $\mathcal{A}$ , who responds with  $s_1$ .
- 3  $\mathcal{A}'$  runs  $\mathcal{A}$  2nd time with the same  $I$ , sends  $r_2 \in \mathbb{Z}_q$  to  $\mathcal{A}$  who responds with  $s_2$ .
- 4 If  $g^{s_1} \cdot h^{-r_1} = I$  and  $g^{s_2} \cdot h^{-r_2} = I$  and  $r_1 \neq r_2$  then output  $x = [(s_1 - s_2) \cdot (r_1 - r_2)^{-1} \bmod q]$ . Else, output nothing.

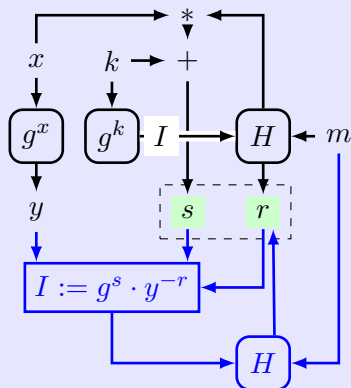


# Schnorr Signature Scheme by Fiat-Shamir Transform

**Fiat-Shamir transform:** signature scheme constructed by letting the signer run the protocol by itself in ROM.

Sign:  $k \leftarrow \mathbb{Z}_q$ ,  
compute  $I := g^k$ ,  
 $r := H(I, m)$ ,  
 $s := [rx + k \bmod q]$   
and output  $(r, s)$

Vrfy: Compute  $I$  and  
output  $1 \iff H(I, m) \stackrel{?}{=} r$ .



NIST published DSS (Digital Signature Standard) which uses Digital Signature Algorithm (DSA, a variant of ElGamal signature scheme), Elliptic Curve Digital Signature Algorithm (ECDSA), and RSA Signature Algorithm.

## Construction 4

- Gen:  $(\mathbb{G}, q, g) \leftarrow \mathcal{G}$ . Two hash functions  $H, F : \{0, 1\}^* \rightarrow \mathbb{Z}_q$ .  
 $x \leftarrow \mathbb{Z}_q$  and  $y := g^x$ .  
 $pk = \langle \mathbb{G}, q, g, y, H, F \rangle$ .  $sk = \langle \mathbb{G}, q, g, x, H, F \rangle$ .
- Sign:  $k \leftarrow \mathbb{Z}_q^*$  and  $r := F(g^k)$ ,  $s := (H(m) + xr) \cdot k^{-1}$ .  
Output  $(r, s)$ .
- Vrfy: Output 1  $\iff r \stackrel{?}{=} F(g^{H(m) \cdot s^{-1}} y^{r \cdot s^{-1}})$ .

Q: Is the verification correct?

## Insecurity

Security of DSS relies on the hardness of discrete log problem. But NO proof of security for DSS based on discrete log assumption.

## The entropy, secrecy and uniqueness of $k$ is critical.

- Case I: If  $k$  is predictable, then  $x$  leaks, since  $s := [(H(m) + xr) \cdot k^{-1} \bmod q]$ , and only  $x$  is unknown;
- Case II: If the same  $k$  is ever used to generate two different signatures under the same  $x$ , then both  $k$  and  $x$  leaks under two signatures.

Q: how?

This attack has been used to learn the private key of the Sony PlayStation (PS3) in 2010.



- 1 Definitions of Digital Signatures
- 2 RSA Signatures
- 3 Digital Signature from The Discrete-Log Problem
- 4 One-Time Signature and Signature from Hash**
- 5 Certificates and Public-Key Infrastructures

# One-Time Signature

**One-Time Signature (OTS):** Under a weaker attack scenario, sign only one message with one secret.

The OTS experiment  $\text{Sigforge}_{\mathcal{A}, \Pi}^{1\text{-time}}(n)$ :

- 1  $(pk, sk) \leftarrow \text{Gen}(1^n)$ .
- 2  $\mathcal{A}$  is given input  $1^n$  and a **single query**  $m'$  to  $\text{Sign}_{sk}(\cdot)$ , and outputs  $(m, \sigma)$ ,  $m \neq m'$ .
- 3  $\text{Sigforge}_{\mathcal{A}, \Pi}^{1\text{-time}}(n) = 1 \iff \text{Vrfy}_{pk}(m, \sigma) = 1$ .

## Definition 5

A signature scheme  $\Pi$  is **existentially unforgeable under a single-message attack** if  $\forall$  PPT  $\mathcal{A}$ ,  $\exists$   $\text{negl}$  such that:

$$\Pr[\text{Sigforge}_{\mathcal{A}, \Pi}^{1\text{-time}}(n) = 1] \leq \text{negl}(n).$$

# Lamport's OTS (1979)

**Idea:** OTS from OWF; one mapping per bit.

## Construction 6

*f is a one-way function.*

■ **Gen:** on input  $1^n$ , for  $i \in \{1, \dots, \ell\}$ :

1 choose random  $x_{i,0}, x_{i,1} \leftarrow \{0, 1\}^n$ .

2 compute  $y_{i,0} := f(x_{i,0})$  and  $y_{i,1} := f(x_{i,1})$ .

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & \cdots & y_{\ell,0} \\ y_{1,1} & y_{2,1} & \cdots & y_{\ell,1} \end{pmatrix} \quad sk = \begin{pmatrix} x_{1,0} & x_{2,0} & \cdots & x_{\ell,0} \\ x_{1,1} & x_{2,1} & \cdots & x_{\ell,1} \end{pmatrix}.$$

■ **Sign:**  $m = m_1 \cdots m_\ell$ , output  $\sigma = (x_{1,m_1}, \dots, x_{\ell,m_\ell})$ .

■ **Vrfy:**  $\sigma = (x_1, \dots, x_\ell)$ , output  $1 \iff f(x_i) = y_{i,m_i}$ , for all  $i$ .

## Theorem 7

*If f is OWF,  $\Pi$  is OTS for messages of length polynomial  $\ell$ .*

# Example of Lamport's OTS

Signing  $m = 011$

$$sk = \begin{pmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{pmatrix} \Rightarrow \sigma = \underline{\hspace{2cm}}$$

$\sigma = (x_1, x_2, x_3)$ :

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & y_{3,0} \\ y_{1,1} & y_{2,1} & y_{3,1} \end{pmatrix} \Rightarrow \begin{array}{l} f(x_1) \stackrel{?}{=} \underline{\hspace{2cm}} \\ f(x_2) \stackrel{?}{=} \underline{\hspace{2cm}} \\ f(x_3) \stackrel{?}{=} \underline{\hspace{2cm}} \end{array}$$

# Proof of Lamport's OTS Security

**Idea:** If  $m \neq m'$ , then  $\exists i^*, m_{i^*} = b^* \neq m'_{i^*}$ . So to forge a signature on  $m$  can invert a single  $y_{i^*, b^*}$  at least.

## Proof.

Reduce  $\mathcal{I}$  inverting  $y$  to  $\mathcal{A}$  attacking  $\Pi$ :

- 1 Construct  $pk$ : Choose  $i^* \leftarrow \{1, \dots, \ell\}$  and  $b^* \leftarrow \{0, 1\}$ , set  $y_{i^*, b^*} := y$ . For  $i \neq i^*$ ,  $y_{i, b} := f(x_{i, b})$ .
- 2  $\mathcal{A}$  queries  $m'$ : If  $m'_{i^*} = b^*$ , stop. Otherwise, return  $\sigma = (x_{1, m'_1}, \dots, x_{\ell, m'_\ell})$ .
- 3 When  $\mathcal{A}$  outputs  $(m, \sigma)$ ,  $\sigma = (x_1, \dots, x_\ell)$ , if  $\mathcal{A}$  output a forgery at  $(i^*, b^*)$ :  $\text{Vrfy}_{pk}(m, \sigma) = 1$  and  $m_{i^*} = b^* \neq m'_{i^*}$ , then output  $x_{i^*, b^*}$ .

$$\Pr[\mathcal{I} \text{ succeeds}] \geq \frac{1}{2\ell} \Pr[\mathcal{A} \text{ succeeds}]$$



# Stateful Signature Scheme

**Idea:** OTS by signing with “**new**” key derived from “**old**” state.

## Definition 8 (Stateful signature scheme)

- **Key-generation** algorithm  $(pk, sk, s_0) \leftarrow \text{Gen}(1^n)$ .  $s_0$  is initial state.
- **Signing** algorithm  $(\sigma, s_i) \leftarrow \text{Sign}_{sk, s_{i-1}}(m)$ .
- **Verification** algorithm  $b := \text{Vrfy}_{pk}(m, \sigma)$ .

### A simple stateful signature scheme for OTS:

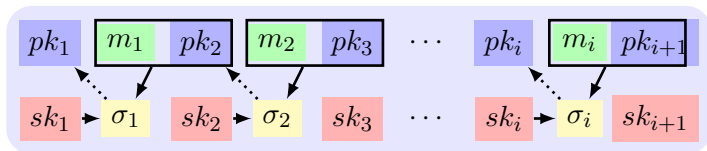
Generate  $(pk_i, sk_i)$  independently, set  $pk := (pk_1, \dots, pk_\ell)$  and  $sk := (sk_1, \dots, sk_\ell)$ .

Start from the state 1, sign the  $s$ -th message with  $sk_s$ , verify with  $pk_s$ , and update the state to  $s + 1$ .

**Weakness:** the upper bound  $\ell$  must be fixed in advance.

# “Chain-Based” Signatures

**Idea:** generate keys “on-the-fly” and sign the key chain.



Use a single public key  $pk_1$ , sign each  $m_i$  and  $pk_{i+1}$  with  $sk_i$ :

$$\sigma_i \leftarrow \text{Sign}_{sk_i}(m_i || pk_{i+1}),$$

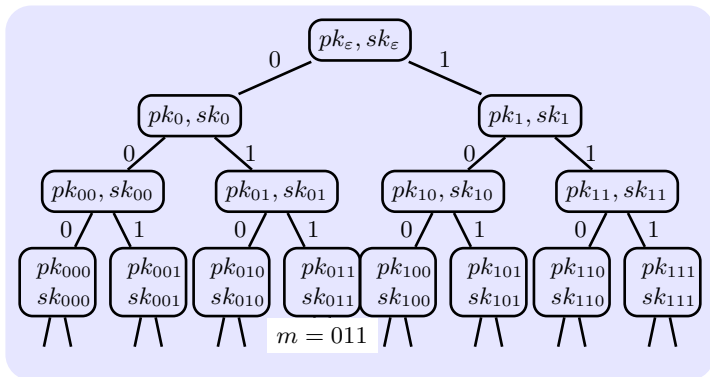
output  $\langle pk_{i+1}, \sigma_i \rangle$ , and verify  $\sigma_i$  with  $pk_i$ .

The signature is  $(pk_{i+1}, \sigma_i, \{m_j, pk_{j+1}, \sigma_j\}_{j=1}^{i-1})$ .

**Weakness:** stateful, not efficient, revealing all previous messages.

# “Tree-Based” Signatures

**Idea:** generate a chain of keys for each message and sign the chain.



- root is  $\epsilon$  (empty string), leaf is a message  $m$ , and internal nodes  $(pk_w, sk_w)$ , where  $w$  is the prefix of  $m$ .
- each node  $pk_w$  “certifies” its children  $pk_{w0}||pk_{w1}$  or  $w$ .



# A Stateless Solution

**Idea:** use deterministic randomness to emulate the state of tree.

Use PRF  $F$  and two keys  $k, k'$  (secrets) to generate  $pk_w, sk_w$ :

- 1 compute  $r_w := F_k(w)$ .
- 2 compute  $(pk_w, sk_w) := \text{Gen}(1^n; r_w)$ , using  $r_w$  as random coins.

$k'$  is used to generate  $r'_w$  that is used to compute  $\sigma_w$ .

## Lemma 9

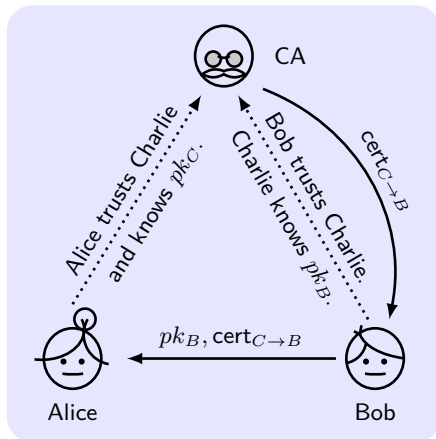
*If OWF exist, then  $\exists$  OTS (for messages of arbitrary length).*

## Theorem 10

*If OWF exists, then  $\exists$  (stateless) secure signature scheme.*

- 1 Definitions of Digital Signatures
- 2 RSA Signatures
- 3 Digital Signature from The Discrete-Log Problem
- 4 One-Time Signature and Signature from Hash
- 5 Certificates and Public-Key Infrastructures**

# Certificates



**Certificates**  $cert_{C \rightarrow B} \stackrel{\text{def}}{=} \text{Sign}_{sk_C}(\text{'Bob's key is } pk_B')$ .

How Alice learn CA's key? How CA learn Bob's key?

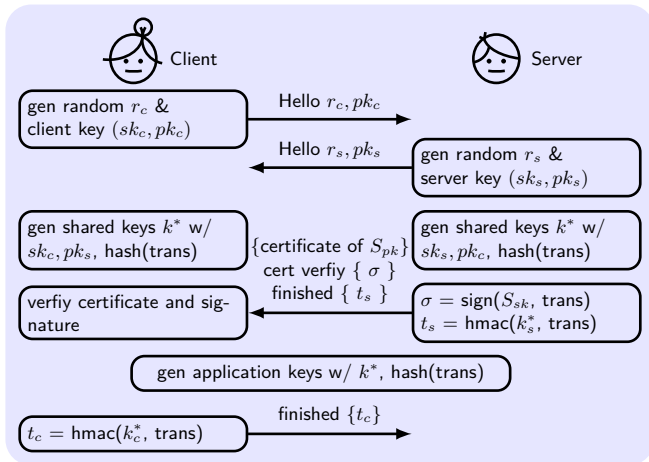
# Public-Key Infrastructure (PKI)

- **A single CA:** is trusted by everybody.
  - Strength: simple
  - Weakness: single-point-of-failure
- **Multiple CAs:** are trusted by everybody.
  - Strength: robust
  - Weakness: cannikin law
- **Delegation and certificate chains:** The trust is transitive.
  - Strength: ease the burden on the root CA.
  - Weakness: difficult for management, cannikin law.
- **“Web of trust”:** No central points of trust, e.g., PGP.
  - Strength: robust, work at “grass-roots” level.
  - Weakness: difficult to manage/give a guarantee on trust.

# TLS 1.3 Handshaking<sup>1</sup>

**Purpose:** client generates secret keys with authenticated server

**Requirements:** the client has the public key of CA, the server has the certificate of its own  $S_{pk}$  issued by CA



<sup>1</sup><https://tls13.ulfheim.net>

# Invalidating Certificates

- **Expiration:** include an *expiry date* in the certificate.

$$\text{cert}_{C \rightarrow B} \stackrel{\text{def}}{=} \text{Sign}_{sk_C}(\text{'bob's key is } pk_B', \text{ date}).$$

- **Revocation:** explicitly revoke the certificate.

$$\text{cert}_{C \rightarrow B} \stackrel{\text{def}}{=} \text{Sign}_{sk_C}(\text{'bob's key is } pk_B', \text{ ###}).$$

“###” represents the serial number of this certificate.

**Cumulated Revocation:** CA generates *certificate revocation list* (CRL) containing the serial numbers of all revoked certificates, signs CRL with the current date.

# Exclusive Ownership

**Exclusive Ownership:** Given any signature generated by a public key  $pk$ , no adversary can find  $pk' \neq pk$  such that the signature can be verified with  $pk'$ .

## Duplicate Signature Key Selection Attack

- Does the validity of a signature with Bob's public key imply that Bob produced the signature with his private key?
- No. For example, the signature  $\sigma$  of  $m$  is generated by Alice with RSA. Bob's key pair is  $(e = 1, d = 1)$  and  $N = \sigma - m$ . Then the signature is also verified with Bob's public key.  
 $\sigma^e \bmod N = \sigma \bmod (\sigma - m) = m$ .
- This attack could be used to cheat [Let's Encrypt] in the ownership of domain name. <sup>2</sup>
- Defence: Check the public key before verification.

---

<sup>2</sup><https://www.ietf.org/mail-archive/web/acme/current/msg00484.html>

**Signcryption:** In a group, each has two pairs of keys:  $(ek, dk)$  for enc, and  $(vk, sk)$  for sig. And all public keys are distributed to everyone. A sender  $S$  and a receiver  $R$  should do to secure both privacy (no other learns  $m$  except  $S$  and  $R$ ) and authenticity ( $R$  is sure about the message is sent from  $S$ ).

## Which signcryption scheme is secure?

- Enc-then-Auth: send  $\langle S, c \leftarrow \text{Enc}_{ek_R}(m), \text{Sign}_{sk_S}(c) \rangle$
- Auth-then-Enc I:  $\sigma \leftarrow \text{Sign}_{sk_S}(m)$ , send  $\langle S, \text{Enc}_{ek_R}(m \parallel \sigma) \rangle$
- Auth-then-Enc II:  $\sigma \leftarrow \text{Sign}_{sk_S}(m \parallel R)$ , send  $\langle S, \text{Enc}_{ek_R}(S \parallel m \parallel \sigma) \rangle$
- Any other method?



- Digital signature provides publicly verifiable authentication and integrity.
- Signatures is related to something only someone knows, which can be publicly verified.
- Signatures are used to convey *trust* from a public key to the data which is signed.