NT219- Cryptography

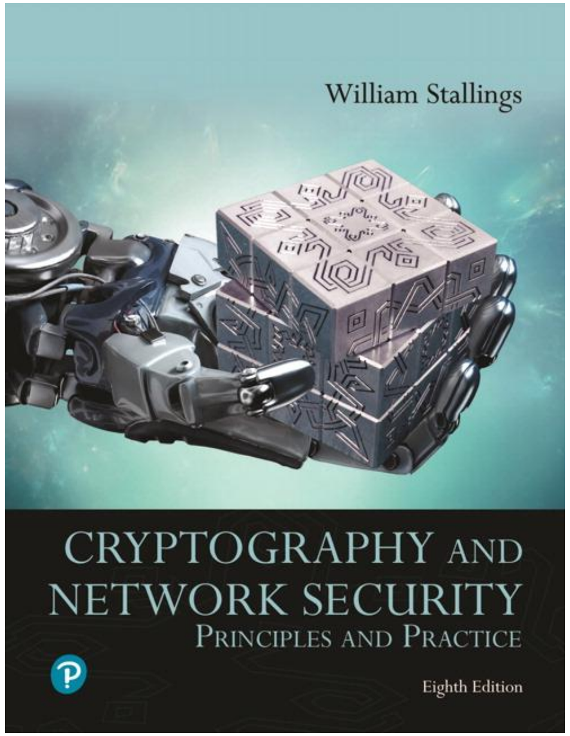
Week 12: Digital Signature

PhD. Ngoc-Tu Nguyen

tunn@uit.edu.vn

Textbooks and References

- Text books



[1] Chapter 13.14

Outline

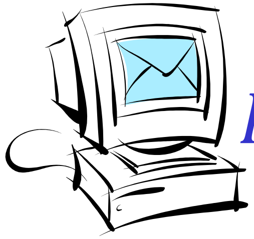
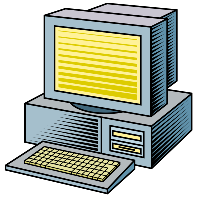
- Motivations
- Overview digital signature process
- Elgamal digital signature scheme
- Schnorr digital signature scheme
- NIST digital signature schemes
  - RSASSA-PKCS, RSASSA-PSS
  - DSA, ECDSA
- Public key distribution (X.509 digital certificates)

Motivations

- How to ensure that the message is the original one?
  - Integrity?
- How to verify that a message comes from the claimed sender?
  - Authentication?

➢ MAC, HMAC

agree a session key K
(using Diffie–Hellman for ex.)

Alice  K $\longleftrightarrow$ K  Bob

$send\ M?$ $\xrightarrow{M, tag = MAC(K,M)(eg. H(K,M))}$ M', tag

$$MAC(K, M') =?\ tag = MAC(K, M)$$

$\Rightarrow M = M'$

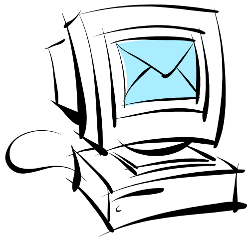
Motivations

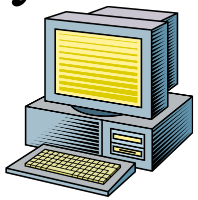
- How to ensure that the message is the original one?
    - Integrity?
- How to verify that a message comes from the claimed sender?
    - Authentication?

➢ MAC, HMAC

If A and B can not agree a session key K?

Alice





Bob

$send\ M?$

$$M, tag = HMAC(K, M) = H(K, M)$$

$$M', tag$$

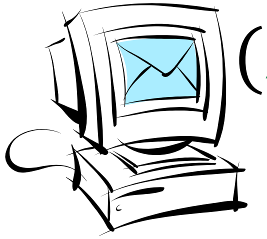
$$H(K, M') = tag = H(K, M)?$$

$$\Rightarrow M = M'$$

Motivations

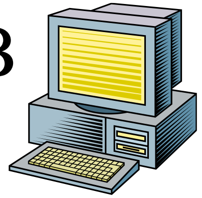
- How to ensure that the message is the original one?
    - Integrity?
- How to verify that a message comes from the claimed sender?
    - Authentication?

➢ **using digital signature**

A  (PK_A, SK_A)

$send\ M?$

$M, tag = sign(SK_A, M)$

B 

D 

C 

PK_A

$\ldots$

M', tag

$verify(PK_A, M', tag)?$

$\Rightarrow M = M'$

Digital Signature Properties

- **Goals:**
  - It must be verifiable the author (signer);
  - It must be verifiable the content integrity and time of the signature;
  - It must be verifiable by **third parties** (to resolve disputes);

Digital signature algorithms

1. Setup system parameters (hash,…)

2. Key generation and distribution: input λ

$SK_A \leftarrow Gen(\lambda); PK_A \leftarrow Gen(\lambda, SK_A)$

3. Signer signs and send out the message

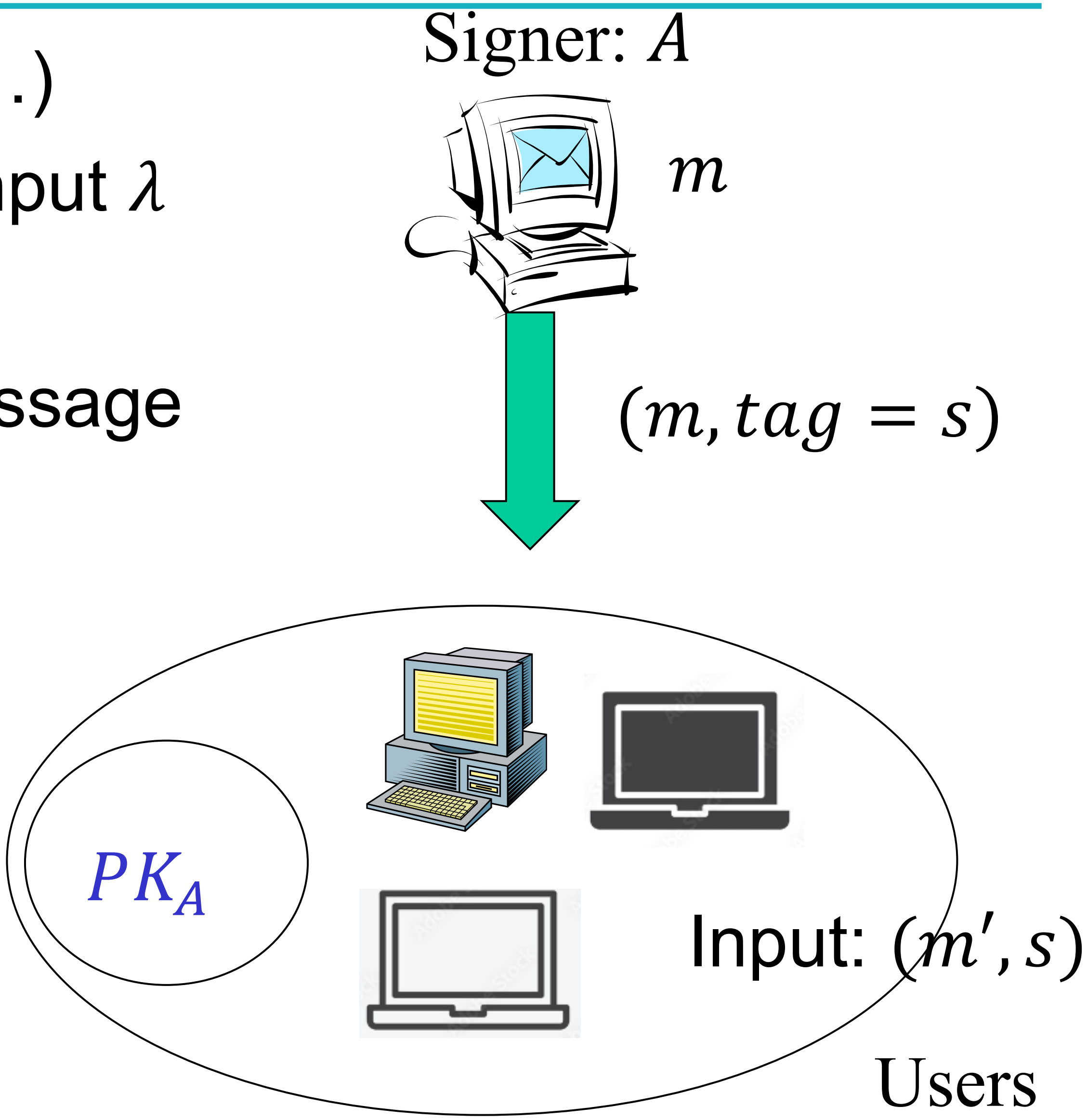
Input: m

sign: $s = sign(m, SK_A)$ → h(m)

Send out: (m, s) (Itext7, QR,…)

4. Users verify message

Verify: $verify(m', s, PK_A)$



Outline

- Motivations
- Overview of the digital signature process
- Elgamal digital signature scheme
- Schnorr digital signature scheme
- NIST digital signature schemes
  - RSASSA-PKCS, RSASSA-PSS
  - DSA, ECDSA
- Public key distribution (X.509 digital certificates)

ElGamal Digital Signature

Parameter generation

- a key length λ;
- λ-bit prime number p;
- a cryptographic hash function H

$$H: \{0,1\}^* \rightarrow \{0,1\}^L \ (L \geq \lambda)$$

- a generator g of the multiplicative group Z_p^*;

Public parameters: $(\boldsymbol{p}, \boldsymbol{g}, \boldsymbol{H})$

Key generation (for signer)

- Secret key: $x \in_R Z_p^*$;
- Public key: $y = g^x \bmod p$;

Key distribution

- Verifiers have to know the signer's public key $y = g^x$;

ElGamal Digital Signature

Signing (the message *m*)

- Choose $k \in_R Z_p^*$
- Compute $r = g^k mode\ p$
- Compute $s = (H(m) - x.r).k^{-1} \bmod (p-1)$
- Output signature $(\boldsymbol{r}, \boldsymbol{s})$, **send out (m, (r,s))**

Verifying a signature

In public key Secret key

- Input (m', r, s)
- Verify $0 < r < p, 0 < s < p - 1$
- Verify $y^r.r^s \bmod p = g^{xr}.g^{k.(H(m)-x.r).k^{-1}}$

$$= g^{H(m)} =?\ g^{H(m')}$$

Outline

- Motivations
- Overview of the digital signature process
- Elgamal digital signature scheme
- Schnorr digital signature scheme
- NIST digital signature schemes
  - RSASSA-PKCS, RSASSA-PSS
  - DSA, ECDSA
- Public key distribution (X.509 digital certificates)

Schnorr Digital Signature

Parameter generation

- prime numbers p,q, where $p = qr + 1$
- a generator g of the multiplicative group Z_p^*;
- a cryptographic hash function H

$$H: \{0,1\}^* \rightarrow Z_q$$

Public parameters: $(\boldsymbol{p}, \boldsymbol{q}, \boldsymbol{g}, \boldsymbol{H})$

Key generation (for signer)

- Secret key: $x \in_R Z_p^*$;
- Public key: $y = g^x \bmod p$;

Key distribution

- Verifiers have to know the signer's public key $y = g^x$;

Schnorr Digital Signature

Signing (the message $\boldsymbol{m}$)

- Choose $k \in_R Z_p^*$
- Compute $r = g^k$
- Compute $e = h(r||m) \in Z_q$
- Compute $s = k - x.e \bmod q = k - x.h(r||m) \bmod q$
- Output signature $(\boldsymbol{s}, \boldsymbol{e})$

Secret key In public key

Verifying a signature

- Input (m', s, e)
- Compute $g^s.y^e \bmod p = g^{k-x.e} . g^{x.e} \bmod p$
  $= g^k = r_v$
- Verify $e = h(r||m) =? h(r_v||m')$

Outline

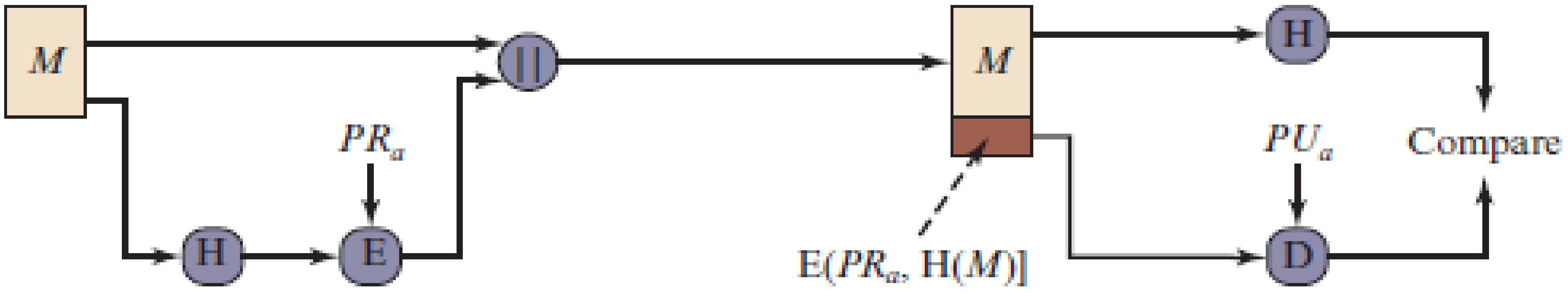
- Motivations
- Overview of the digital signature process
- Elgamal digital signature scheme
- Schnorr digital signature scheme
- NIST digital signature schemes
  - RSASSA-PKCS, RSASSA-PSS
  - DSA, ECDSA
- Public key distribution (X.509 digital certificates)

NIST Digital Signature Algorithm

- Published by NIST as Federal Information Processing Standard FIPS 186 (1994)
  - https://nvlpubs.nist.gov/nistpubs/Legacy/FIPS/fipspub186.pdf
  - Makes use of the Secure Hash Algorithm (SHA)
- The latest version, FIPS 186-4 (2013)
  - https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.186-4.pdf
- Current version FIPS 186-5 (2023)
  - https://csrc.nist.gov/publications/detail/fips/186/5/final

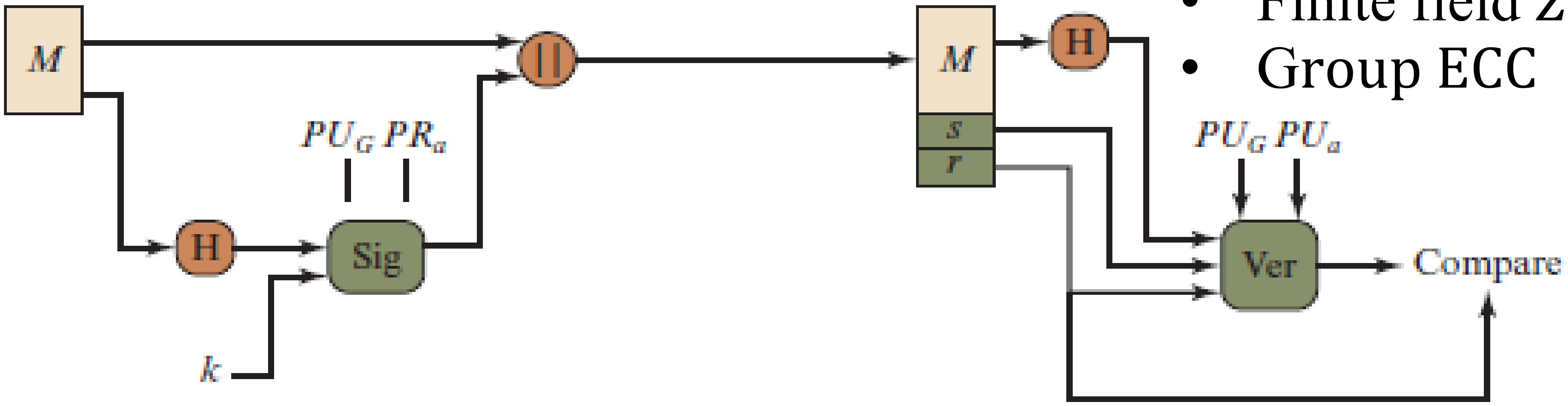
FIPS 203, 204 (2024)

Two Approaches to Digital Signatures



(a) RSA approach

- Finite field Z_p
- Group ECC



(b) DSA approach

RSASSA Signatures

Raw version

- Public key is (n, e), private key is d
- To sign message m: $s = hash(m)^d \bmod n$
  - Signing and encryption are the same mathematical operation in RSA
- To verify signature s on message m':

  $s^e \bmod n = (hash(m)^d)^e \bmod n = hash(m)$? = $hash(m')$
  - Verification and dencryption are the same mathematical operation in RSA
- Message must be padded and hashed (why?)

RSASSA Signatures

- Padding m
  - Public-Key Cryptography Standards PKCS #1(v2.2):

    RSASSA-PKCS
    - https://datatracker.ietf.org/doc/html/rfc8017
  - Probabilistic Signature Scheme

    RSASSA-PSS
    - https://datatracker.ietf.org/doc/html/rfc8017

RSASSA-PKCS

PKCS padding

(M, emLen)

EM = 0x00 || 0x01 || 0xff… 0xff || 0x00 || DigestInfo value ||H(M) ⇒ RSA encrypt

$signature: s$

MD2: (0x)30 20 30 0c 06 08 2a 86 48 86 f7 0d 02 02 05 00 04 10 || H(M)

MD5: (0x)30 20 30 0c 06 08 2a 86 48 86 f7 0d 02 05 05 00 04 10 || H(M)

SHA-1: (0x)30 21 30 09 06 05 2b 0e 03 02 1a 05 00 04 14 || H(M)

SHA-224: (0x)30 2d 30 0d 06 09 60 86 48 01 65 03 04 02 04 05 00 04 1c || H(M)

SHA-256: (0x)30 31 30 0d 06 09 60 86 48 01 65 03 04 02 01 05 00 04 20 || H(M)

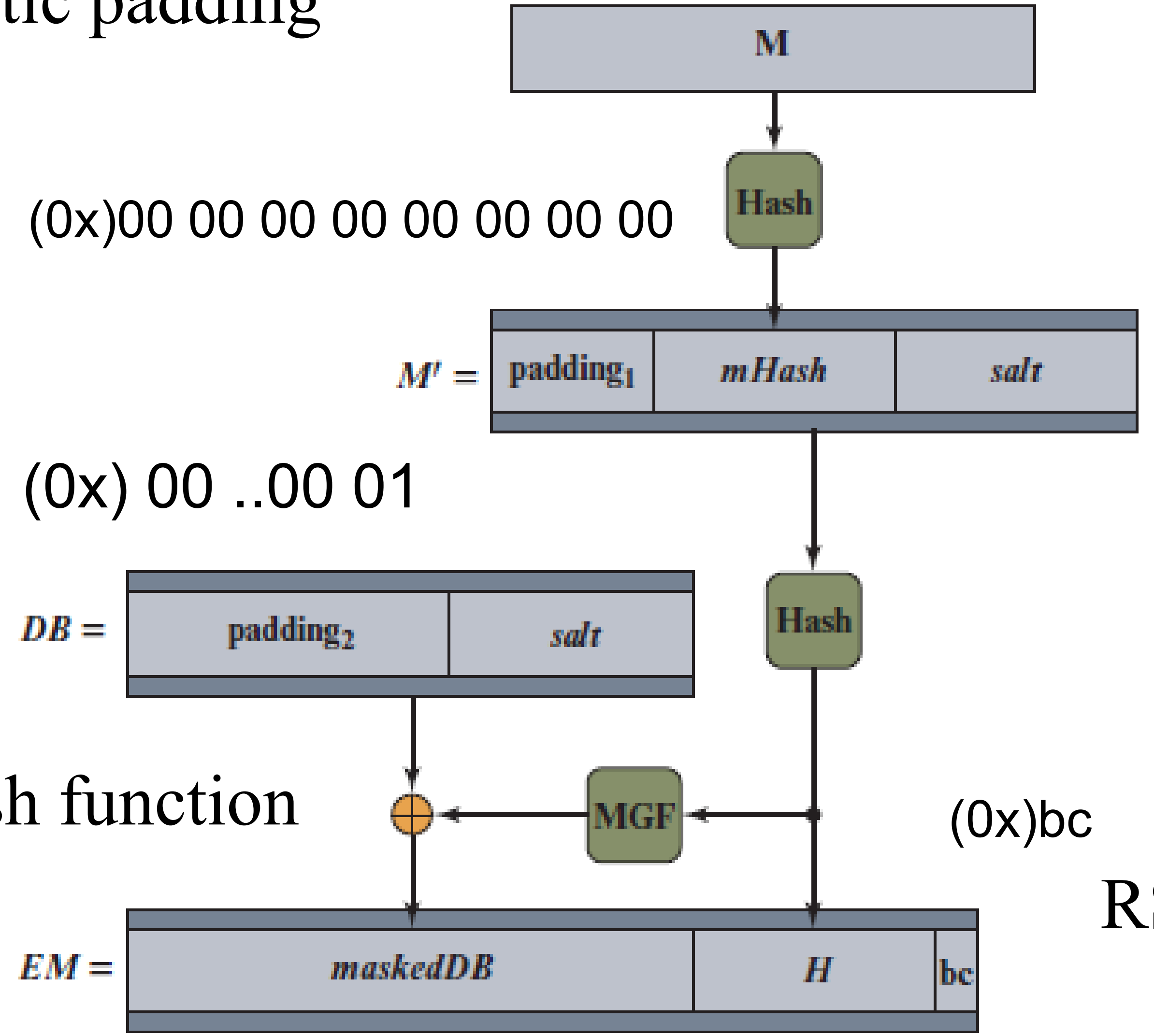
SHA-384: (0x)30 41 30 0d 06 09 60 86 48 01 65 03 04 02 02 05 00 04 30 || H(M)

SHA-512: (0x)30 51 30 0d 06 09 60 86 48 01 65 03 04 02 03 05 00 04 40 || H(M)

SHA-512/224: (0x)30 2d 30 0d 06 09 60 86 48 01 65 03 04 02 05 05 00 04 1c || H(M)

RSASSA-PSS Encoding

Probabilistic padding



MGF: a hash function

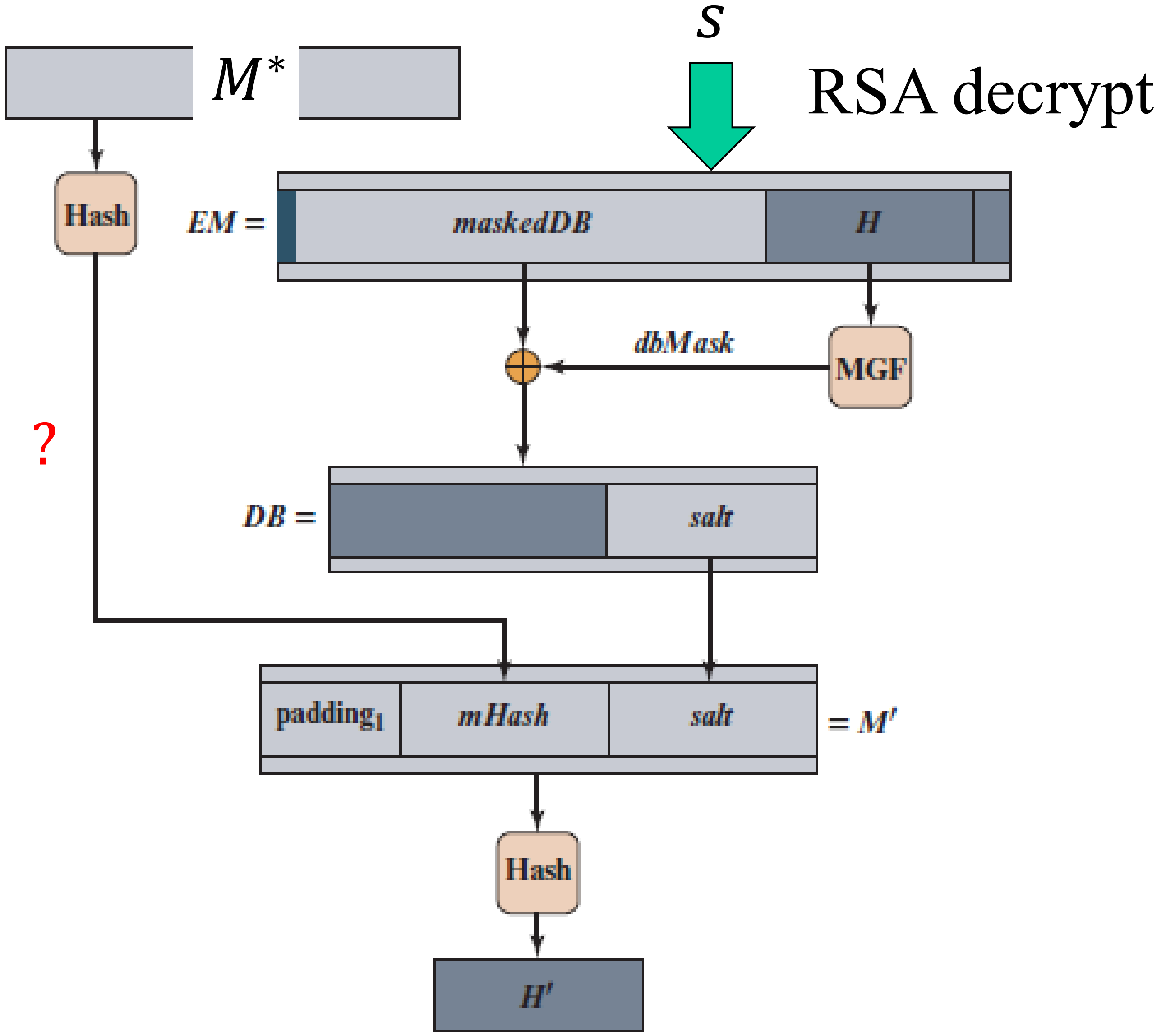
RSA encrypt

$signature\text{: } s$

RSASSA-PSS verifying

(M^*, s)

H(M*)=?H(M)=mHash



Outline

- Motivations
- Overview of the digital signature process
- Elgamal digital signature scheme
- Schnorr digital signature scheme
- NIST digital signature schemes
  - RSASSA-PKCS, RSASSA-PSS
  - DSA,ECDSA
- Public key distribution (X.509 digital certificates)

The Digital Signature Algorithm (DSA)

- DSA Parameters

p a prime modulus, where $2^{L-1} < p < 2^L$, and L is the bit length of p. Values for L are provided in Section 4.2.

q a prime divisor of $(p - 1)$, where $2^{N-1} < q < 2^N$, and N is the bit length of q. Values for N are provided in Section 4.2.

g a generator of a subgroup of order q in the multiplicative group of GF(p), such that $1 < g < p$.

x the private key that must remain secret; x is a randomly or pseudorandomly generated integer, such that $0 < x < q$, i.e., x is in the range $[1, q-1]$.

y the public key, where $y = g^x \bmod p$.

k a secret number that is unique to each message; k is a randomly or pseudorandomly generated integer, such that $0 < k < q$, i.e., k is in the range $[1, q-1]$.

The Digital Signature Algorithm (DSA)

DSA Parameters

$2^{L-1} < p < 2^L, p-1 \vdots q$

$2^{N-1} < q < 2^N$

$\langle g \rangle = GF(p) = \mathbb{Z}_p$

$H: \{0,1\}^* \to \{0,1\}^l; l \geq \min(L, N)$

$L = 1024, N = 160$

$L = 2048, N = 224$

$L = 2048, N = 256$

$L = 3072, N = 256$

Key generation (for signer)

Secret key: $x \in_R [1, q-1]$

Public key: y= $g^x \bmod p \in \mathbb{Z}_p$

Key distribution

- Verifiers have to know the signer's public key $y = g^x \; mod \; p$;

The Digital Signature Algorithm (DSA)

Signing (the message $\boldsymbol{m}$)

- Choose secret for each message: $k \in_R [1, q-1]$
- Compute $r = (g^k mod\ p)\ mod\ q$
- Compute $s = k^{-1}(H(m) + x.r) \bmod q$
- Output signature (r, s)

The Digital Signature Algorithm (DSA)

Signing (the message *m*)

- (m, r, s)

$r = (g^k mod\, p)\, mod\, q$
$s = k^{-1}(H(m) + x.r) \bmod q$

Verifying a signature

In public key   Secret key

- Input (m', r, s)
- Compute:

$w = s^{-1}\, mod\, q = (H(m) + x.r)^{-1}.k \bmod q$

$u_1 = H(m').w \bmod q$

$=H(m')(H(m) + x.r)^{-1}.k$

$u_2 = r.w = r.(H(m) + x.r)^{-1}.k \bmod q$

- Verify $v = g^{u_1}.y^{u_2} = g^{u_1}.g^{x.u_2} = g^{u_1 + x.u_2}$

$= g^{k(H(m') + x.r)((H(m) + x.r)^{-1})} =?\, r$

Outline

- Motivations
- Overview of the digital signature process
- Elgamal digital signature scheme
- Schnorr digital signature scheme
- NIST digital signature schemes
  - RSASSA-PKCS, RSASSA-PSS
  - DSA,ECDSA
- Public key distribution (X.509 digital certificates)

Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA parameters

- Prime number: p $(or\ f(x))$
- Curve coefficients: $a, b \in \mathbb{Z}_p$
- Base points: $G \in E(\mathbb{Z}_p)$
- The number $n = ord(\langle G \rangle)$
- The number $h = \dfrac{ord\,(E(\mathbb{Z}_p))}{n}$
- $H: \{0,1\}^* \to \{0,1\}^l,\ l = l(n)$

Key generation (for signer)

- Secret key: $d \in_R [1, n-1]$
- Public key: Q= $d.G \in E(\mathbb{Z}_p)$

Key distribution: Curve, Q

Bit length of n	Maximum Cofactor (h)	Comparable Security Strength
224 - 255	2^{14}	approximately $n/2$; at least 112 bits
256 - 383	2^{16}	approximately $n/2$; at least 128 bits
384 - 511	2^{24}	approximately $n/2$; at least 192 bits
≥ 512	2^{32}	approximately $n/2$; at least 256 bits

NIST.FIPS.186-5

Elliptic Curve Digital Signature Algorithm (ECDSA)

Signing (the message $\boldsymbol{m}$)

- Choose secret for each message: $k \in_R [1, n-1]$
- Compute $R = k.G = (x_1, y_1), r = x_1$
- Compute $s = k^{-1}(H(m) + d.r) \bmod n$
- Output signature (r, s) //len(p) +len(n)

Elliptic Curve Digital Signature Algorithm (ECDSA)

Signing (the message m)

- (m, r, s)

$r = x_1 \text{ where } kG = (x_1, y_1)$
$s = k^{-1}(H(m) + d.r) \bmod n$

Verifying a signature

- Input $(m', r, s), PK$,
- Compute:

In public key Secret key

$w = s^{-1} \bmod q = (H(m) + d.r)^{-1}.k \bmod n$
$u_1 = H(m').w \bmod n = H(m')(H(m) + d.r)^{-1}.k \bmod n$
$u_2 = r.w = r.(H(m) + d.r)^{-1}.k \bmod n$
$v_x = u_1 G + u_2 Q = (u_1 + du_2)G$
$= k(H(m') + d.r)\left((H(m) + d.r)^{-1}\right)G = (x_1', y_1')$

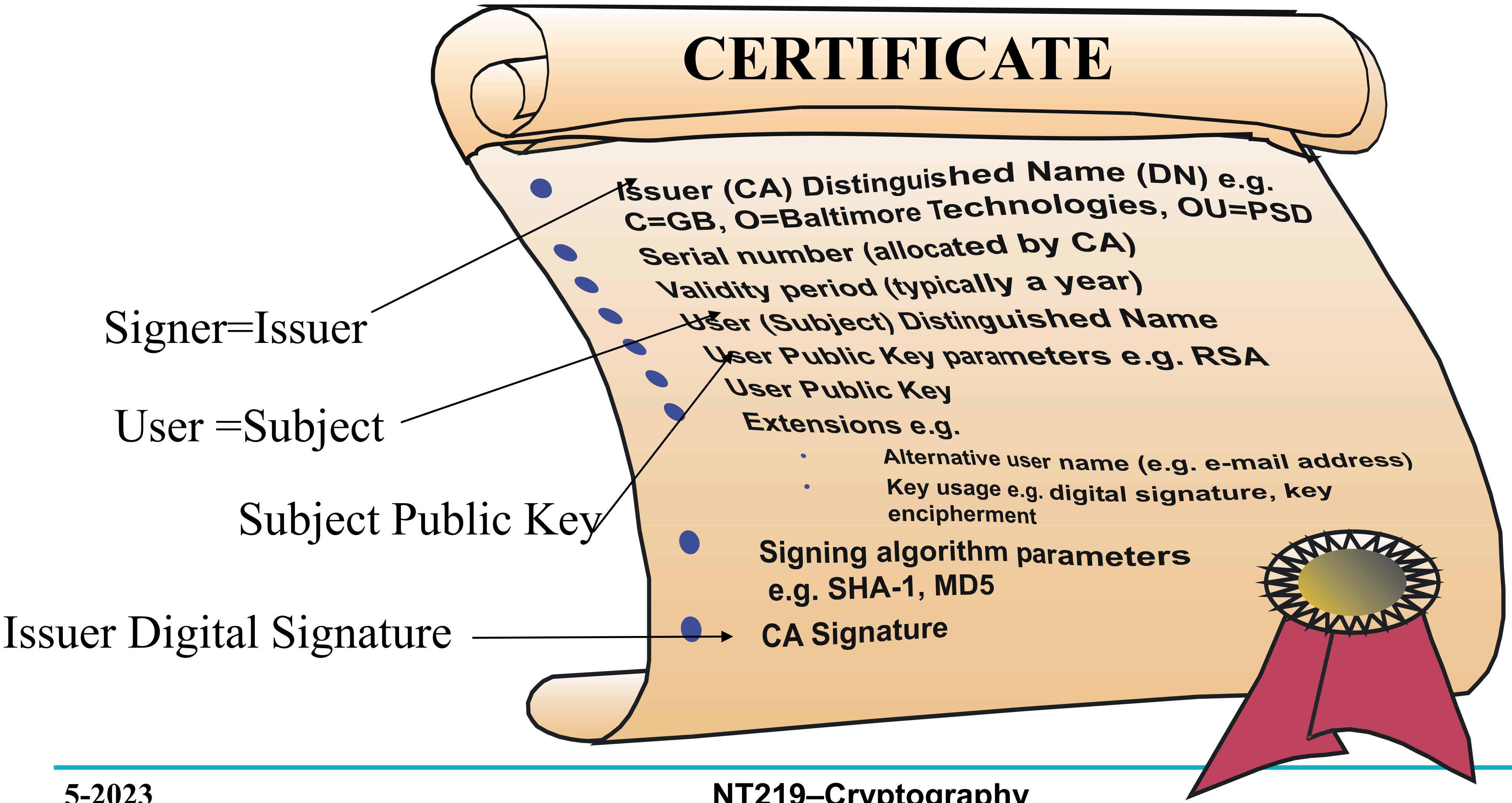
- Verify $v_x = x_1' =?\, r$

Outline

- Motivations
- Overview of the digital signature process
- Elgamal digital signature scheme
- Schnorr digital signature scheme
- NIST digital signature schemes
  - RSASSA-PKCS
  - RSASSA-PSS
  - ECDSA
- Public key distribution (X.509 digital certificates)

X.509 digital certificate

$$\text{Certificate} = (M = \{user, public\ key, \dots\}, s)$$



X.509 digital certificate

Problems

How are Digital Certificates Issued?

Who is issuing them?

Why should I Trust the Certificate Issuer (Signers)?

How can I check if a Certificate is valid?

How can I revoke a Certificate?

Who is revoking Certificates?

X.509 digital certificate Formats

- *Version*: which version the certificate is using
- *Serial number*: a unique # assigned to the certificate within the same CA
- *Algorithm*: name of the hash function and the public-key encryption algorithm
- *Issuer*: name of the issuer
- *Validity period*: time interval when the certificate is valid
- *Subject*: name of the certificate owner
- *Public key*: subject's public-key and parameter info.
- *Extension*: other information (only available in version 3)
- *Properties*: encrypted hash value of the certificate using $K_{CA}{}^{r}$

Public Key Infrastructure (PKI)

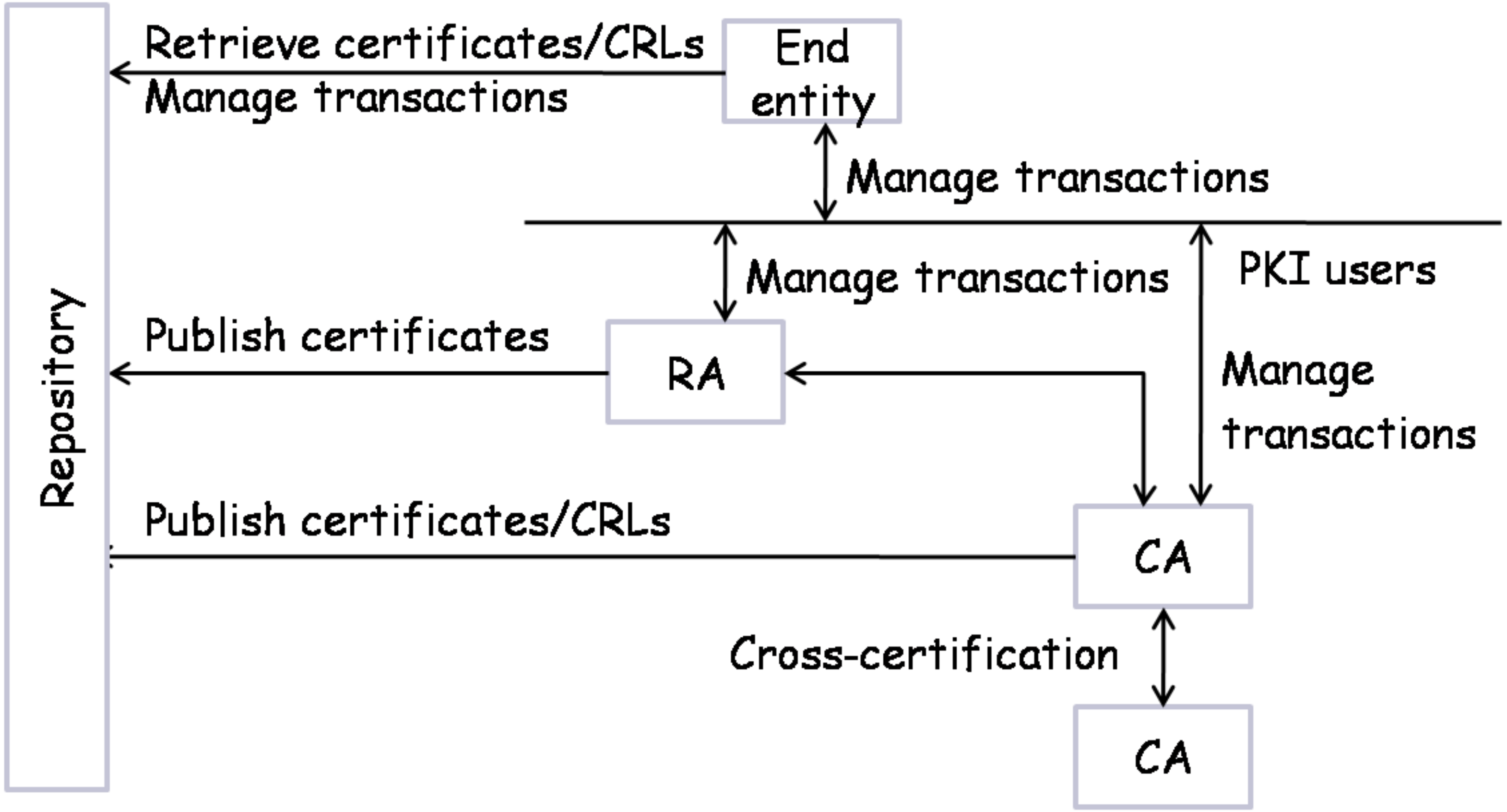
- PKI is a mechanism for using PKC
- PKI issues and manages subscribers' public-key certificates and CA networks (signers):
  - Determine users' legitimacy
  - Issue public-key certificates upon users' requests
  - Extend public-key certificates' valid time upon users' requests
  - Revoke public-key certificates upon users' requests or when the corresponding private keys are compromised
  - Store and manage public-key certificates
  - Prevent digital signature singers from denying their signatures
  - Support CA networks to allow different CAs to authenticate public-key certificates issued by other CAs

X.509 PKI (PKIX)

- Recommended by IETF
- Four basic components:
  1. end entity (users, verifyers)
  2. certificate authority (CA): responsible of issuing and revoking public-key certificates;
  3. registration authority (RA): verifying identities of owners of public-key certificates
  4. Repository: responsible of storing and managing public-key certificates

https://datatracker.ietf.org/doc/html/rfc5280
(2008, updated 2013, 2018)

PKIX Architecture



Transaction managements:

- Registration
- Initialization
- Certificate issuing and publication
- Key recovery
- Key generation
- Certificate revocation
- Cross-certification

X.509 digital certificate

Example: check digital certificate of "facebook.com"

- **Check server certificate**

```
echo | openssl s_client -servername www.facebook.com -connect www.facebook.com:443 2> 1.txt | openssl x509 -text
```

- **Dowload server cert**

```
echo | openssl s_client -servername www.facebook.com -connect www.facebook.com:443 2>1 | openssl x509 -out facebook.cer -text
```

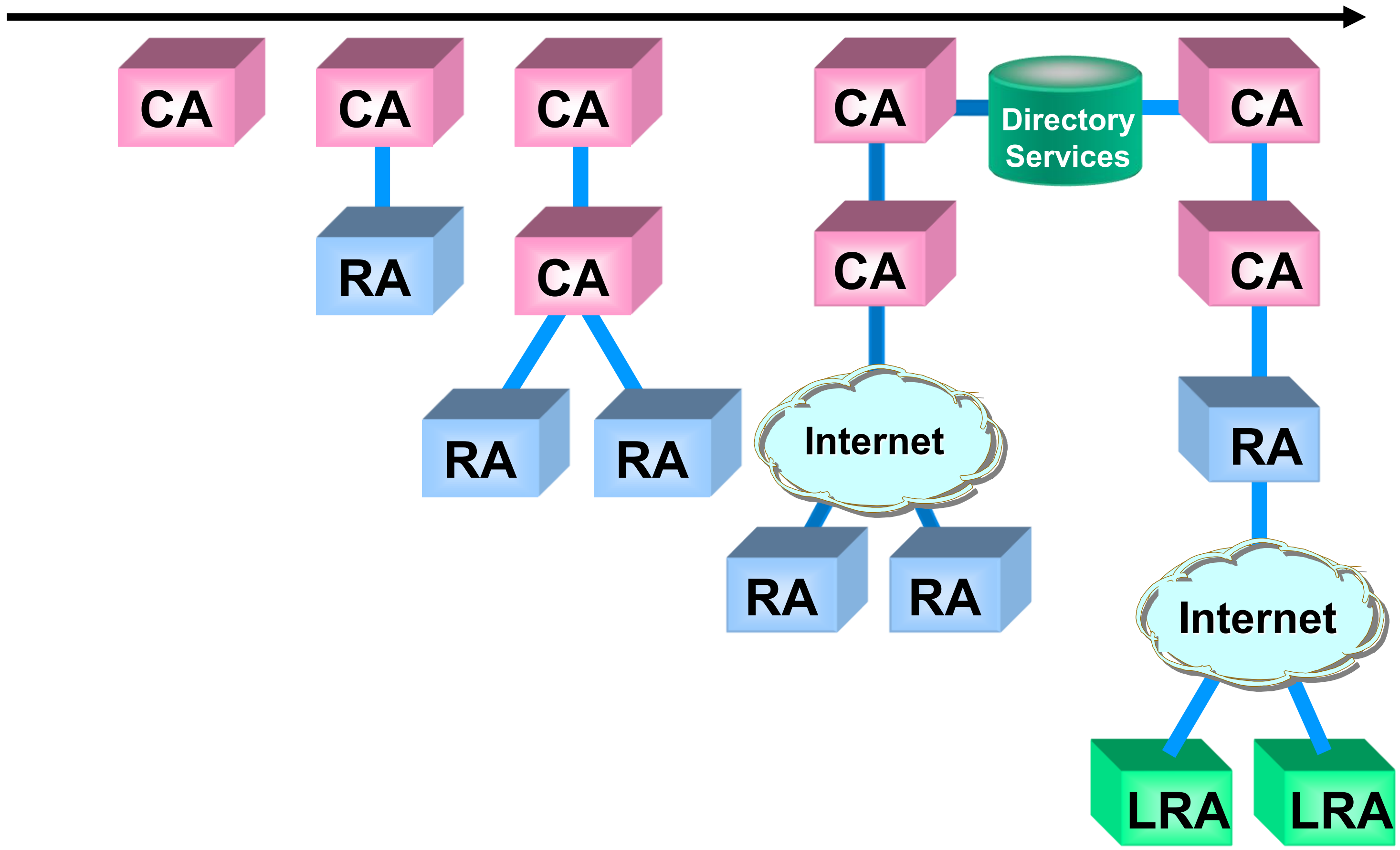
- **Check server cert file**

```
openssl x509 -in facebook.cer -inform pem -text -noout
```

https://www.openssl.org/docs/man1.1.1/man1/openssl-s_client.html

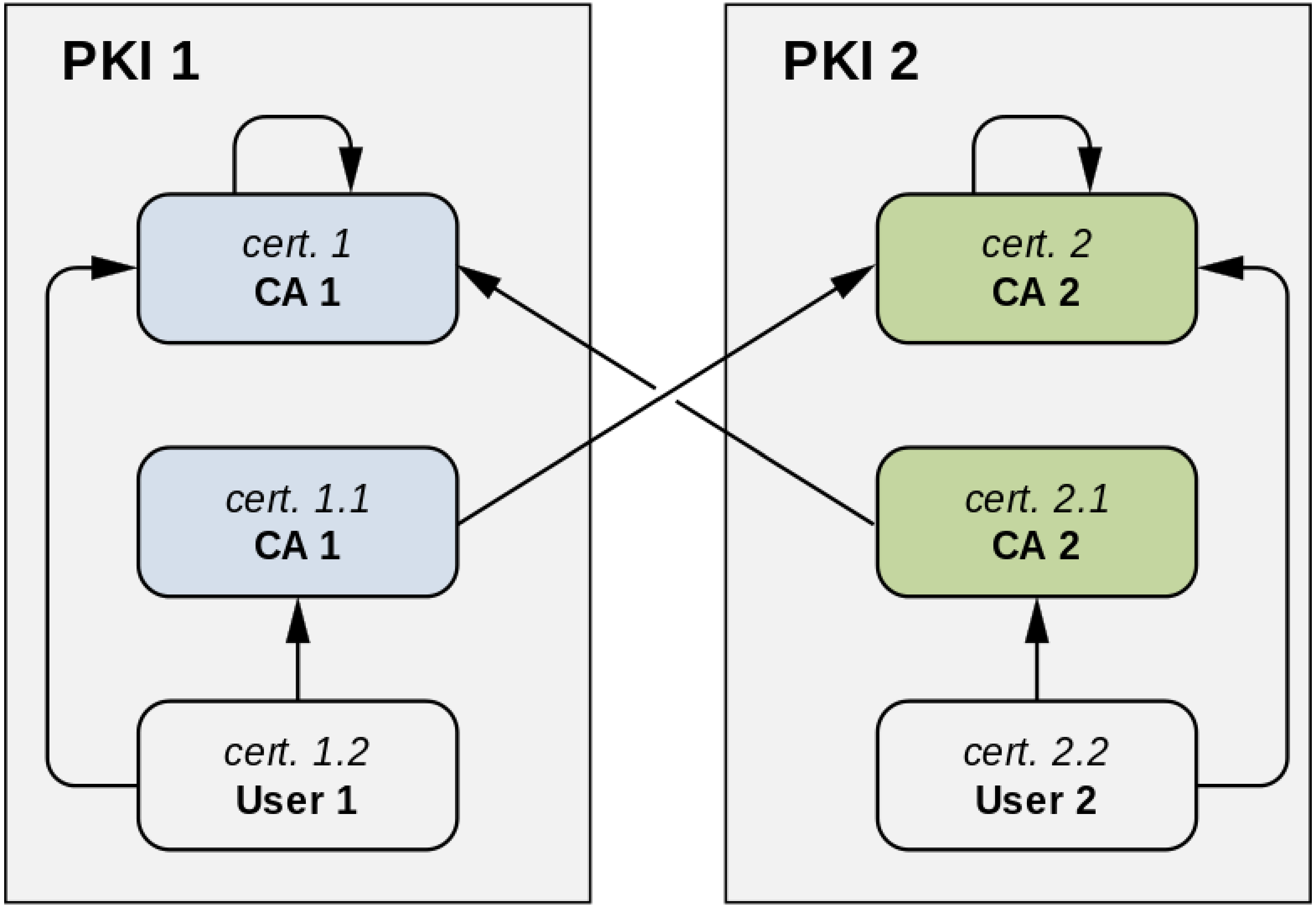
https://www.openssl.org/docs/man1.1.1/man1/openssl-x509.html

CA Technology Evolution



openssl s_client -connect facebook.com:443 -showcerts

Cross-certification between two PKIs



Attack terminology on digital signature

- **Key-only attack**
  - Attackers only knows signer's public key
- **Known message attack**
  - Attackers is given access to a set of messages and their signatures

Attack terminology on DS

- **Generic chosen message attack**
  - Attackers chooses a list of messages before attempting to break signer's signature scheme (independent of signer's public key);
  - Attackers then obtains from signer valid signatures for the chosen messages;

Attack terminology on DS

- **Directed chosen message attack**
  - Similar to the generic attack, except that the list of messages to be signed is chosen after attackers knows signer's public key but before any signatures are seen;
- **Adaptive chosen message attack**
  - attackers may request from signer signatures of messages that depend on previously obtained message-signature pairs;

Forgery attacks

- **Total break**
  - Attackers determines signer's private key;
- **Universal forgery**
  - Attackers finds an efficient signing algorithm that provides an equivalent way of constructing signatures on arbitrary messages;
- **Selective forgery**
  - Attackers forges a signature for a particular message chosen by signer;
- **Existential forgery**
  - Attackers forges a signature for at least one message; Attackers has no control over the message;

Digital Signature Requirements

- The signature must be a bit pattern that depends on the message being signed;
- The signature must use some information unique to the sender to prevent both forgery and denial;
- It must be relatively easy to produce the digital signature;
- It must be relatively easy to recognize and verify the digital signature;
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message;
- It must be practical to retain a copy of the digital signature in storage;

Direct Digital Signature

- **Direct Digital signature scheme** involves only the communicating parties
  - It is assumed that the destination knows the public key of the source;
- **Confidentiality**: encrypting the entire message + signature with a shared secret key
  - It is important to perform the signature function first and then an outer confidentiality function
  - In case of dispute some third party must view the message and its signature;
- **The validity of the scheme** depends on the security of the sender’s private key
  - If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature;
  - One way to thwart or at least weaken this ploy is to require every signed message to include a timestamp and to require prompt reporting of compromised keys to a central authority