Lecture 11

DIGITAL SIGNATURES AND CERTIFICATES

R. Rhouma

UTAS
Sultanate of Oman

September 2022

CSSY2201 : Introduction to Cryptography

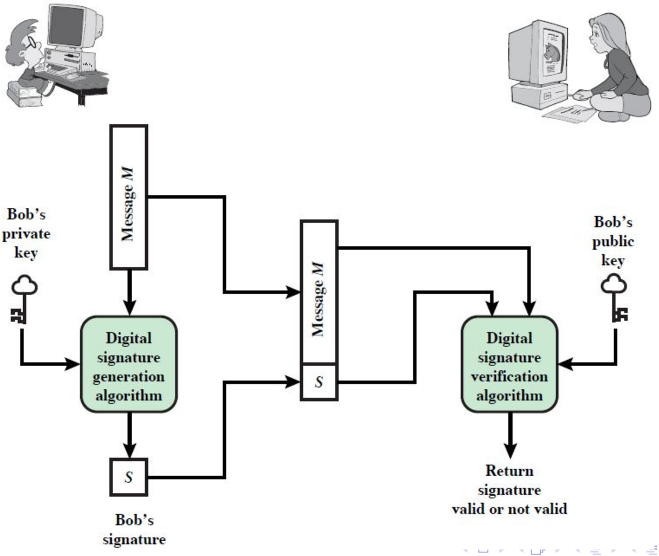
Plan

1. DigitalSignature
2. The DSA algorithm of the DSS standard recommended by NIST
3. Distribution of public keys in asymmetric algs

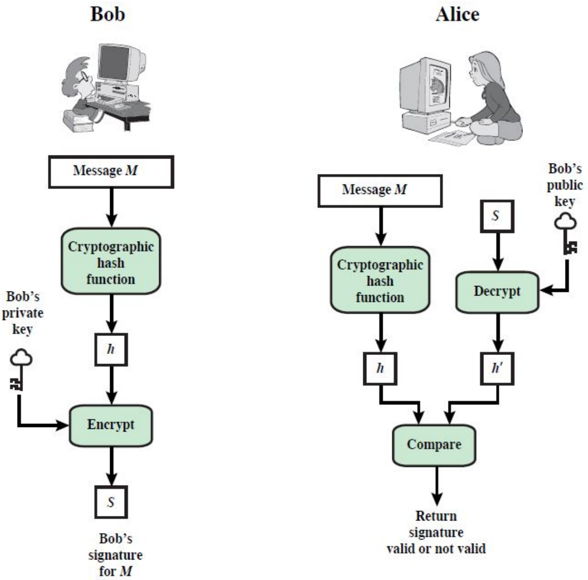
Digital Signature

- Similar to MAC
- the message digest is encrypted by the message sender's private key
- Anyone who knows the sender's public key can verify the integrity of the message
- a hacker who wants to modify the message needs to know the sender's private key
- 3 properties :
  - it must verify the author, time and date of the signed document
  - it must authenticate the content at the time of signing
  - it must be verified by a third party to resolve disputes

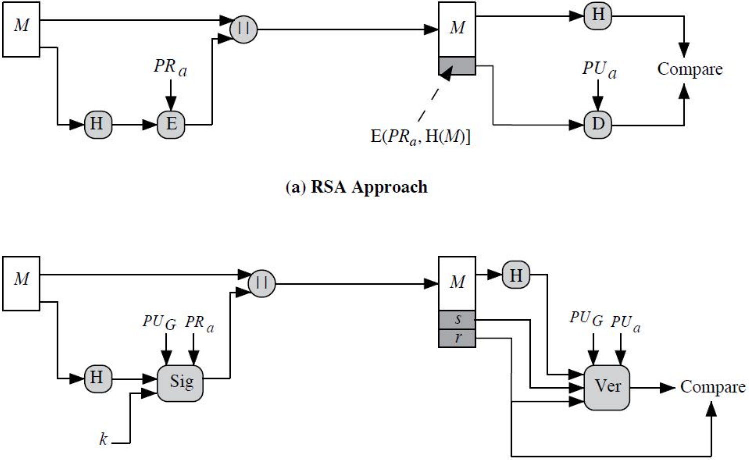
General model of the digital signature


Bob's private key
Message M
Digital signature generation algorithm
S
Bob's signature
Message M
S
Bob's public key
Digital signature verification algorithm
Return signature valid or not valid

Detailed model of the digital signature



2 approaches of digital signature



(a) RSA Approach

(b) DSS Approach

NIST DSA

- DSA uses SHA
- creates a 320-bit size signature
- smaller and faster than RSA
- is a digital signature alg only (no encryption, no key sharing)
- security related to the discrete logarithm problem
- is a variant of ElGamal

Key generation in DSA

- Given a shared public key (p,q,g) :
  - choose a 160-bit prime q
  - choose a large prime nb p such that $2^{L-1} < p < 2^{L}$
    - with L= 512 to 1024 bits and is a multiple of 64
    - so that q is a 160 bit prime divisor of (p-1)
  - choose $g = h^{(p-1)/q}$
    - with $1 < h < p - 1$ and $h^{(p-1)/q}$ $mod\ p > 1$
- users choose a private key & calculate a public key :
  - choose a random private key : $x < q$
  - calculates a public key : $y = g^{x}$ $mod\ p$

Creating the DSA signature

- To sign a M message, the sender :
    - generates a signing key k with $k < q$
    - the nb k must be random, and must be destroyed after use and never reused
- Calculate the signature formed by the pair (r, s) :

$$r = (g^k \, mod \, p) \, mod \, q$$

$$s = [k^{-1}(H(M) + x \times r)] \, mod \, q$$

- send signature (r, s) with message M

DSA signature verification

- The receiver receives the message M and its signature (r, s)
- to verify the signature, the receiver does the following :

$$w = s^{-1} \; mod \; q$$

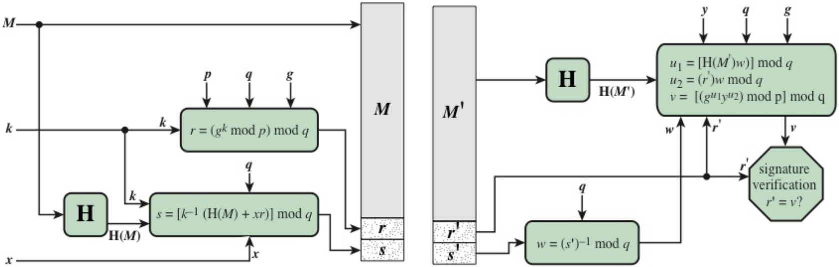
$$u_1 = [H(M) \times w] \; mod \; q$$

$$u_2 = (r \times w) \; mod \; q$$

$$v = [(g^{u_1} \times y^{u_2}) \; mod \; p] \; mod \; q$$

- If $v = r$, then the signature is verified

DSA generation and verification : schemes



(a) Signing

(b) Verifying

Public key distribution

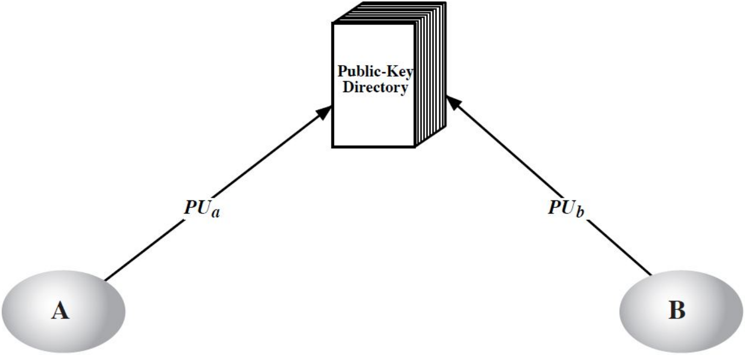
Can be done by :

- public announcement
- through a publicly available archive (directory)
- through A public key authority
- Public key certificates

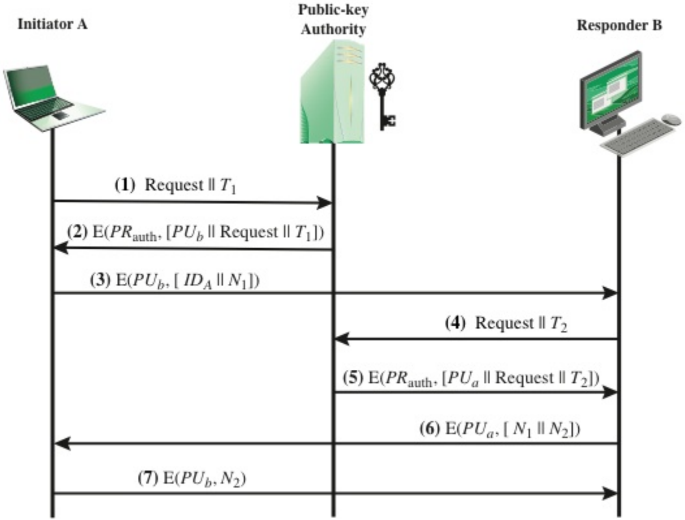
Distribution by Public Announcement

- Users distribute their public keys to (interested) recipients or by broadcast to the community
  - attach PGP public keys to emails, or send them to new groups or broadcast to address book mails
- Main problem of this method is modification = counterfeit
  - anyone can create a key pretending to be someone else and distribute it
  - until the tampering is discovered, the adversary can communicate as if it were the legitimate user

Distribution of keys by publicly available directory



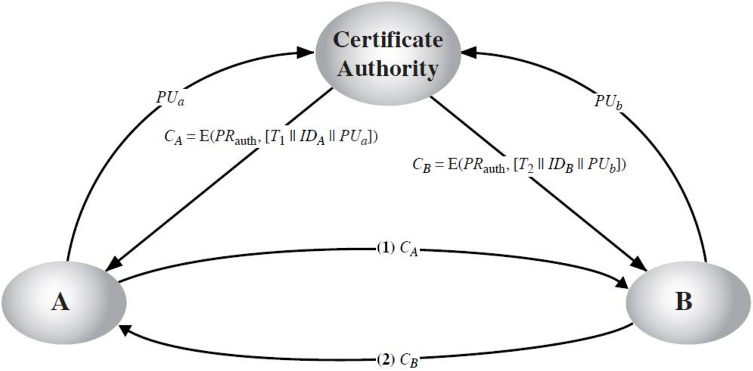
Key distribution by public key authority


Initiator A
Public-key Authority
Responder B
(1) Request ‖ T_1
(2) E(PR_{auth}, [PU_b ‖ Request ‖ T_1])
(3) E(PU_b, [ID_A ‖ N_1])
(4) Request ‖ T_2
(5) E(PR_{auth}, [PU_a ‖ Request ‖ T_2])
(6) E(PU_a, [N_1 ‖ N_2])
(7) E(PU_b, N_2)

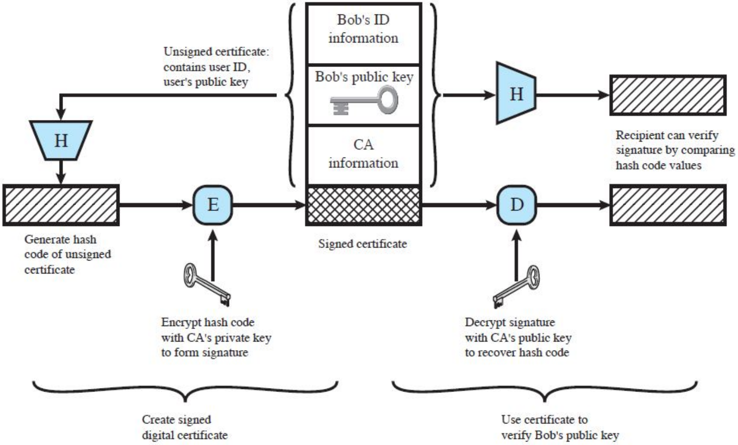
Key distribution by Certificates

- Certificates allow the exchange of public keys without real-time access to a public key authority
- The certificate links identity to public key
  - usually with other info such as validity period, usage rights
- certificate content is signed by a public key authority or certificate authority (CA)
- any user who knows the CA public key can verify the signature

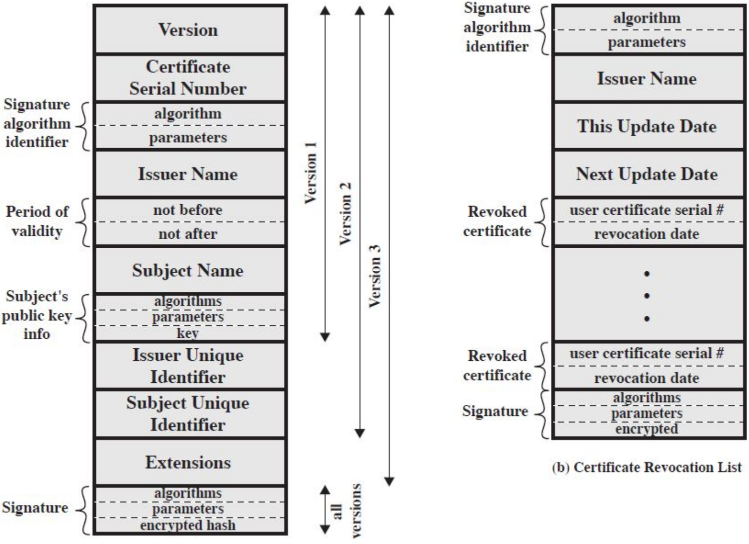
Creation and exchange of certificates


Certificate Authority
PU_a
PU_b
$C_A = E(PR_{auth}, [T_1 \| ID_A \| PU_a])$
$C_B = E(PR_{auth}, [T_2 \| ID_B \| PU_b])$
(1) C_A
(2) C_B
A
B

Use of the certificate


Bob's ID information
Bob's public key
CA information
Unsigned certificate: contains user ID, user's public key
H
H
E
D
Generate hash code of unsigned certificate
Signed certificate
Recipient can verify signature by comparing hash code values
Encrypt hash code with CA's private key to form signature
Decrypt signature with CA's public key to recover hash code
Create signed digital certificate
Use certificate to verify Bob's public key

Certificate Format X.509



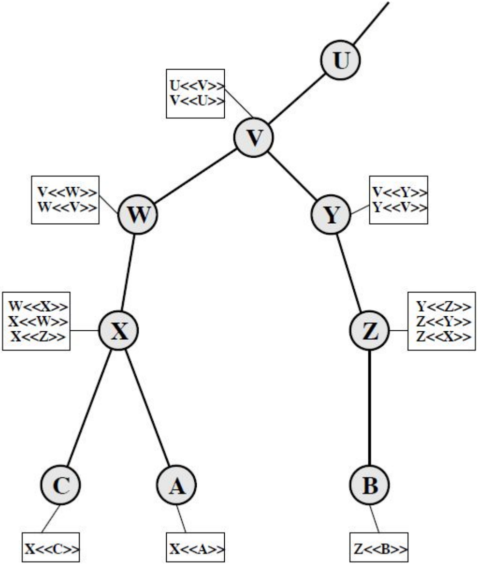
(b) Certificate Revocation List

(a) X.509 Certificate

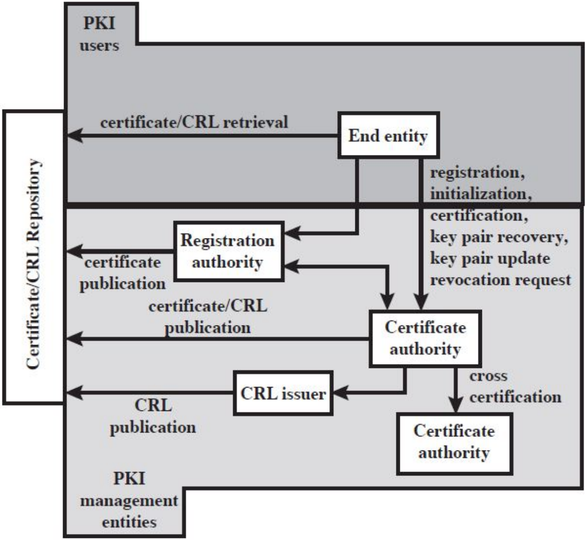
The hierarchy of certification authorities

- If both users share the same CA, then they both know its public key
- otherwise the certification authorities form a hierarchy
- use certificates binding hierarchy members to validate other CAs
- each CA has certificates for its clients and parents
- every customer trusts his parents
- hierarchy allows verification of any CA certificate by users of other CAs in the hierarchy

The hierarchy of certification authorities



PKI : Public Key Infrastructure


PKI users
certificate/CRL retrieval
End entity
registration, initialization, certification, key pair recovery, key pair update revocation request
Certificate/CRL Repository
Registration authority
certificate publication
certificate/CRL publication
Certificate authority
CRL issuer
cross certification
CRL publication
Certificate authority
PKI management entities

management of certificates by PKI

The functions of PKI :

- registration
- initialization
- certification
- key pair recovery
- update key pair
- revocation request
- solution for cross certification
- protocols : CMP, CMC