IT 422 Network Security

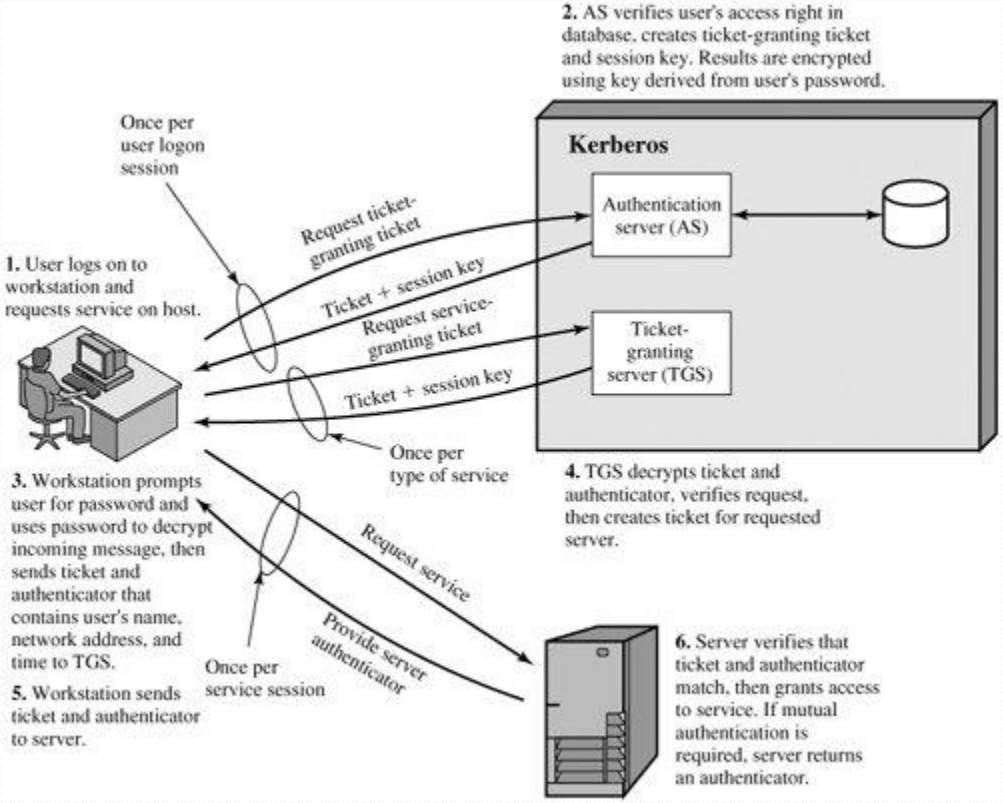
Authentication – X509 - PKI

Yasser F. O. Mohammad

REMINDER 1: How to Authenticate

- What you know.
  - Password/passphrase
- What you have.
  - Smart Cards
- What you are: Static Biometrics.
  - Fingerprint/Face recognition
- What you do: Dynamic Biometrics.
  - Handwriting characteristics

REMINDER 2: Overview of Kerberos 4

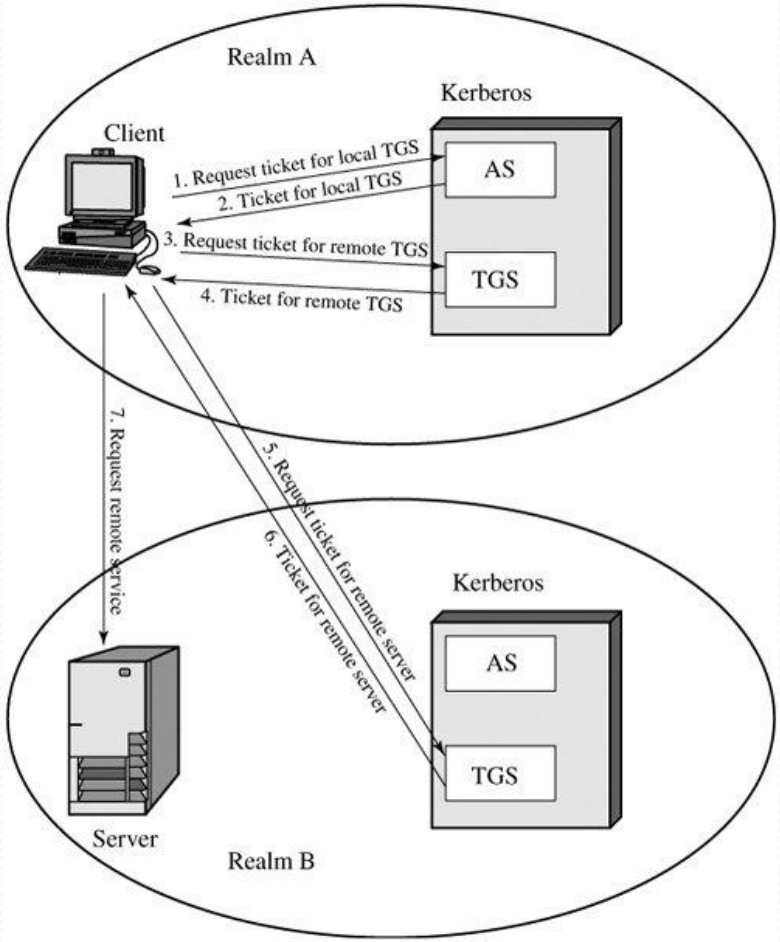


REMINDER 3: Multiple Realm Authentication

- Each Kerberos server much share a key with each other Kerberos server (in version 4)

- In summary:

  Get A TGT from your local TGS for the TGS of the other realm, then use this ticket to request tickets in services in the other realm



REMINDER 4:Kerberos 5 Exchange

(1) C ⟶ AS Options||ID_c||$Realm_c$||ID_{tgs}||$Times$||$Nonce_1$

(2) AS ⟶ C $Realm_c$||ID_C||$Ticket_{tgs}$||E(K_c, [$K_{c,tgs}$||$Times$||$Nonce_1$||$Realm_{tgs}$||ID_{tgs}])

$Ticket_{tgs}$ = E(K_{tgs}, [$Flags$||$K_{c,tgs}$||$Realm_c$||ID_c||AD_c||$Times$])

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C ⟶ TGS Options||ID_v||$Times$|||$Nonce_2$||$Ticket_{tgs}$||$Authenticator_c$

(4) TGS ⟶ C $Realm_c$||ID_c||$Ticket_v$||E($K_{c,tgs}$, [$K_{c,v}$||$Times$||$Nonce_2$||$Realm_v$||ID_v])

$Ticket_{tgs}$ = E(K_{tgs}, [$Flags$||$K_{C,tgs}$||$Realm_c$||ID_C||AD_C||$Times$])

$Ticket_v$ = E(K_v, [Flags||$K_{c,v}$||$Realm_c$||ID_C||AD_c||Times])

$Authenticator_c$ = E($K_{c,tgs}$, [ID_C||$Realm_c$||TS_1])

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C ⟶ V Options||$Ticket_v$||$Authenticator_c$

(6) V ⟶ C $E_{K_{c,v}}$[TS_2||Subkey||Seq#]

Ticket$_v$ = E(K_v, [Flags||$K_{c,v}$||$Realm_c$||ID_C||AD_C||Times])

$Authenticator_c$ = E($K_{c,v}$, [ID_C||$Realm_c$||TS_2||$Subkey$||$Seq\#$])

(c) Client/Server Authentication Exchange to obtain service

(1) C ⟶ AS ID_c||ID_{tgs}||TS_1

(2) AS ⟶ C E(K_c,[$K_{c,tgs}$||ID_{tgs}||TS_2||$Lifetime_2$||$Ticket_{tgs}$])

$Ticket_{tgs}$ = E(K_{tgs}, [$K_{c,tgs}$||ID_c||AD_c||ID_{tgs}||TS_2||$Lifetime_2$])

(a) Authentication Service Exchange to obtain ticket-granting ticket

(3) C ⟶ TGS ID_v||$Ticket_{tgs}$||$Authenticator_c$

(4) TGS ⟶ C E($K_{c,tgs}$, [$K_{c,v}$||ID_v||TS_4||$Ticket_v$])

$Ticket_{tgs}$ = E(K_{tgs}, [$K_{c,tgs}$||ID_C||AD_C||ID_{tgs}||TS_2||$Lifetime_2$])

$Ticket_v$ = E(K_v, [$K_{c,v}$||ID_C||AD_C||ID_v||TS_4||$Lifetime_4$])

$Authenticator_c$ = E($K_{c,tgs}$, [ID_C||AD_C||TS_3])

(b) Ticket-Granting Service Exchange to obtain service-granting ticket

(5) C ⟶ V $Ticket_v$||$Authenticator_c$

(6) V ⟶ C E($K_{c,v}$, [TS_5 + 1]) (for mutual authentication)

$Ticket_v$ = E(K_v, [$K_{c,v}$||ID_c||AD_c||ID_v||TS_4||$Lifetime_4$])

$Authenticator_c$ = E($K_{c,v}$,[ID_c||AD_C||TS_5])

(c) Client/Server Authentication Exchange to obtain service

Kerberos version 4

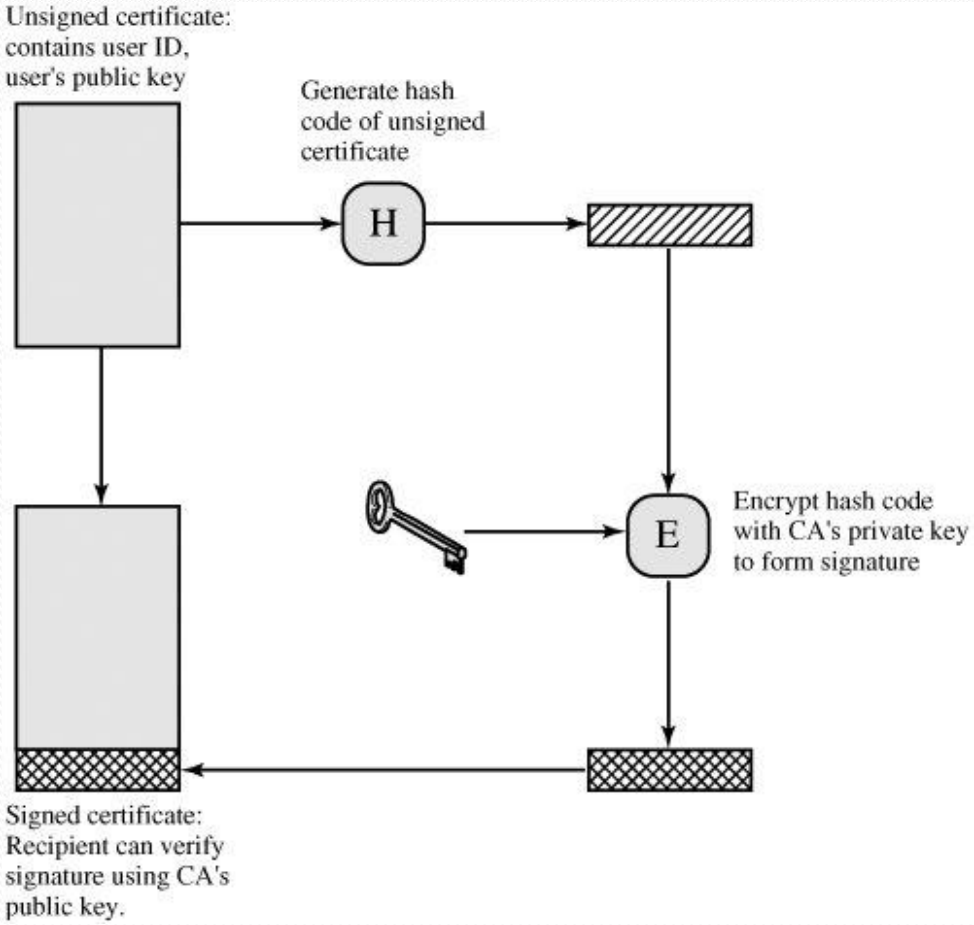
What is X.509?

- Part of X.500 standard for directory services
- Recommended by ITU-T in 1988
- Used in many applications
  - SSL/TLS
  - S/MIME
  - IP Security
  - SET
  - etc
- No specific public key algorithm but usually RSA

Requirements for Certificates

- Any user who knows the public key of the CA should be able to verify the public key of a certificate holder.
- No one can modify the certificate except the CA.

How certificates are signed?

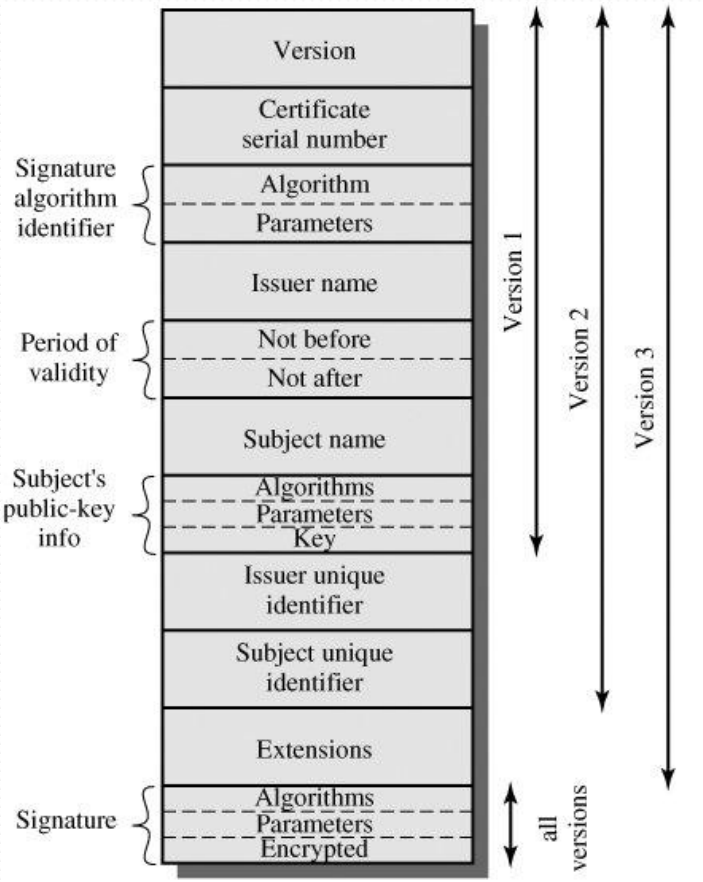


X.509 Certificate Format

CA<<A>> = CA {V, SN, AI, CA, T_A, A, Ap}

where

Y <<X>>	= the certificate of user X issued by certification authority Y
Y {I}	= the signing of I by Y. It consists of I with an encrypted hash code appended



(a) X.509 certificate

Limitation of single CA

- Each user must get the public key of the CA securely.
- In most cases multiple CAs exist and some users know the public key of each of them

Problem with multiple CAs

- A is registered with the CA X_1
  - A Knows Public key of the CA named X_1
  - X_1<<A>> exists
- B is registered with the CA X_2
  - B Knows Public key of the CA named X_2
  - X_2<<B>> exists
- Now if A gets X_2<<B>>, it cannot use it

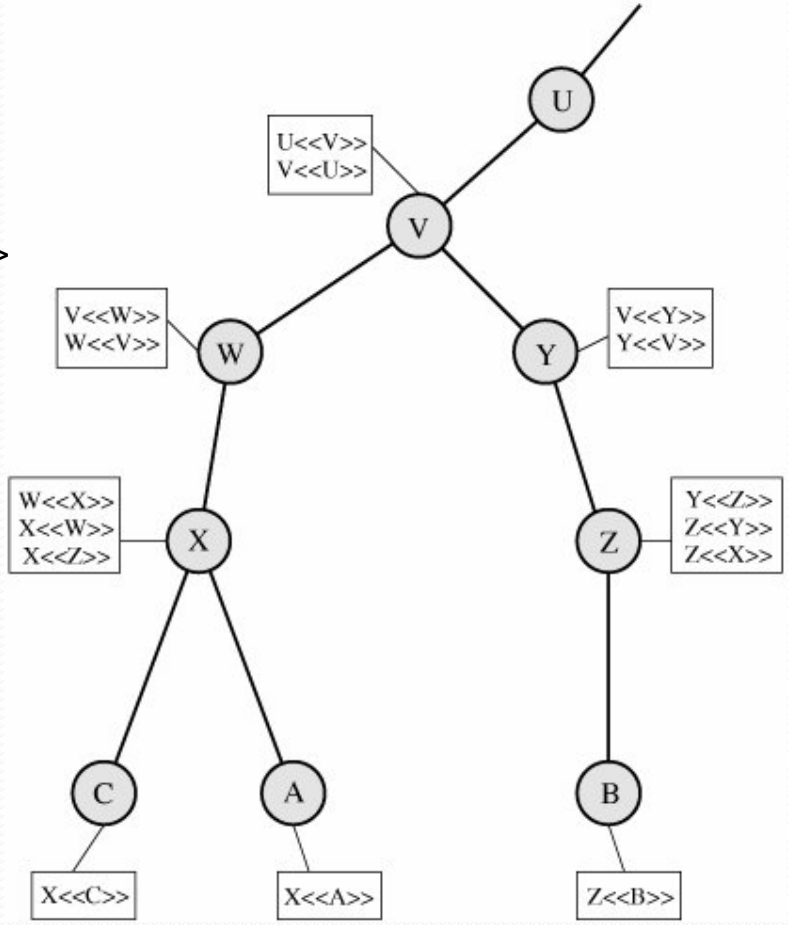
Solution to multiple CAs problem

- X_1 issues a certificate for X_2 : $X_1\langle\langle X_2\rangle\rangle$
- A receives $X_2\langle\langle B\rangle\rangle$
  - A asks X_2 for $X_1\langle\langle X_2\rangle\rangle$
  - A gets the public key of X_2 from this certificate
  - A uses PU_{X2} to decrypt $X_2\langle\langle B\rangle\rangle$
  - Now A can verify B's certificate
- In general
  - $X_1\langle\langle X_2\rangle\rangle$ $X_2\langle\langle X_3\rangle\rangle$ $X_3\langle\langle X_4\rangle\rangle$ ..... $X_{N-1}\langle\langle X_N\rangle\rangle X_N\langle\langle B\rangle\rangle$

Example CA hierarchy

B can get the public key of A using the chain:

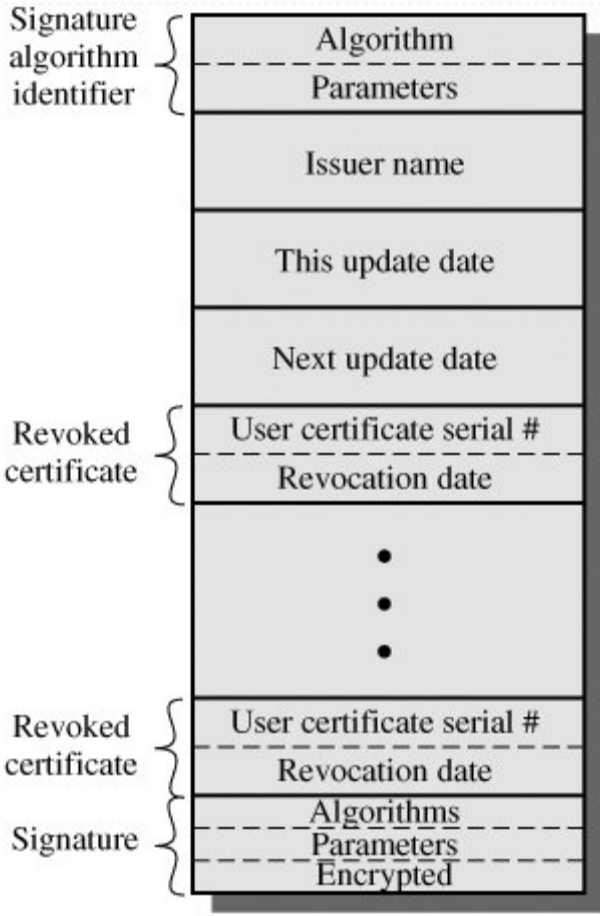
Z<<Y>> Y <<V>> V <<W>> W <<X>>X <<A>>



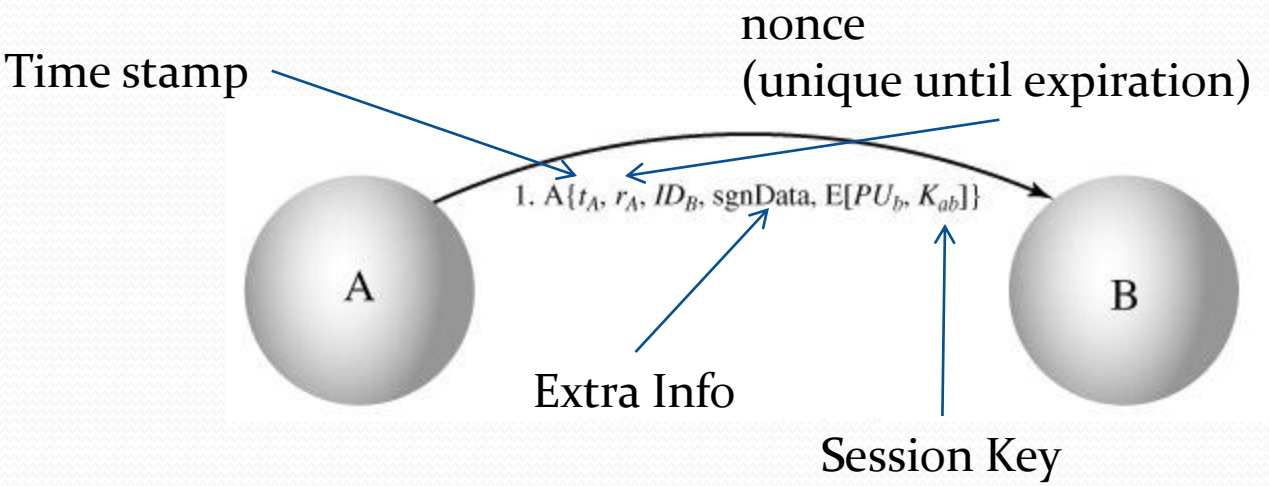
Revoking certificates

1. The Private Key was compromised.
2. CA no longer sure that the user is whom he is supposed to be.
3. The CA itself was compromised.

- *Each CA keeps a list of all its certificates that are revoked.*
- *Each user should check with the CA each time (s)he gets a certificate.*
- *Each user should keep a local list of revoked certificates (to reduce delays)*



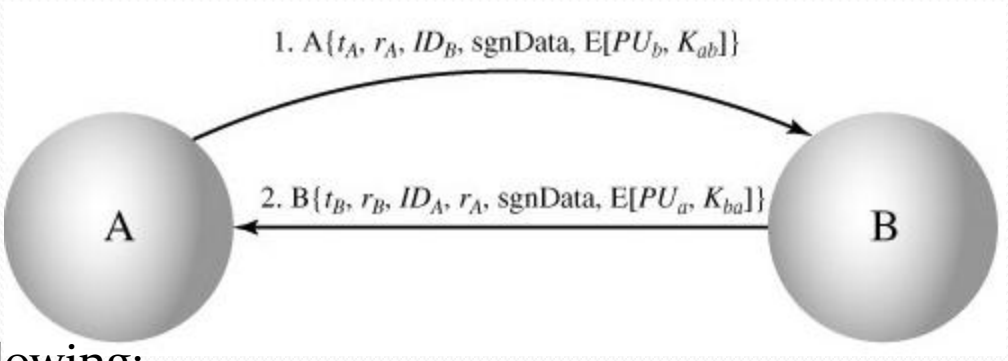
One Way Authentication



Confirms the following:

1. The identity of A and that the message was generated by A
2. That the message was intended for B
3. The integrity and originality (it has not been sent multiple times) of the message

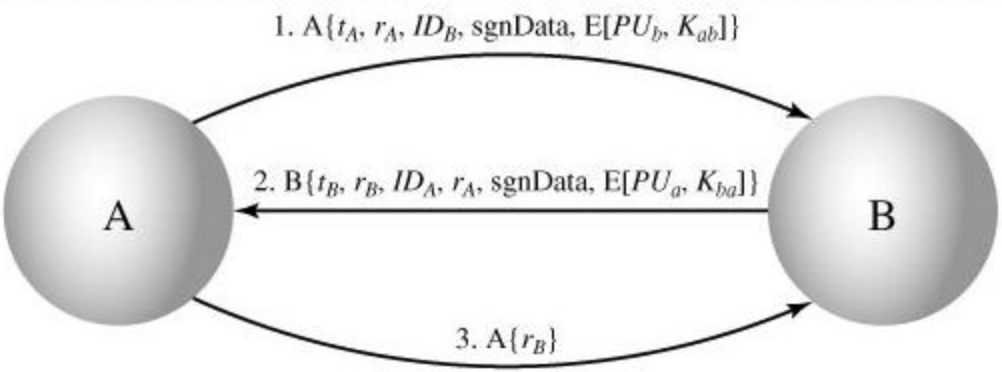
Two Way Authentication



Confirms the following:

1. The identity of A and that the message was generated by A
2. That the message was intended for B
3. The integrity and originality (it has not been sent multiple times) of the message
4. The identity of B and that the reply message was generated by B
5. That the message was intended for A
6. The integrity and originality of the reply

Three way Authentication



Confirms the same six things as in two way authentication but does not require synchronized clocks

X.509 version 3

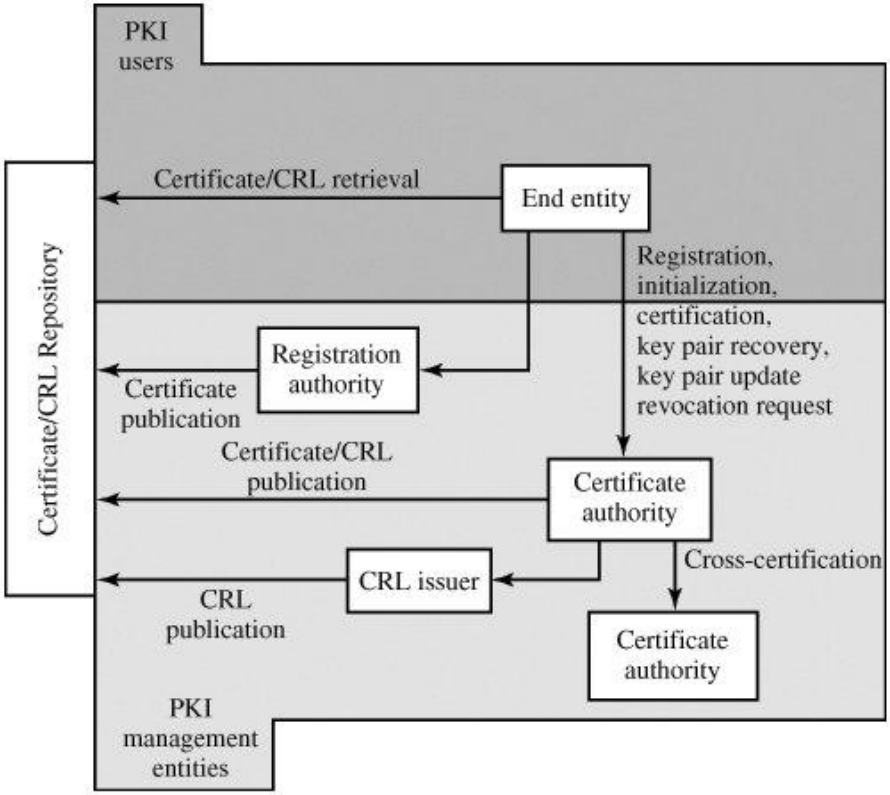
- SELF READ

PKI

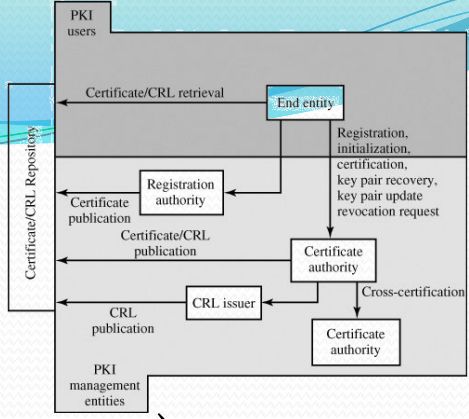
- Public-key infrastructure (PKI): the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography

PKIX

- Designed by IETF
- Based on X.509 certificates



PKIX



- **End entity:** End users, devices (e.g., servers, routers), etc
- **Certification authority (CA):** The issuer of certificates and (usually) certificate revocation lists (CRLs).
- **Registration authority (RA):** The RA is often associated with the End Entity registration process.
- **CRL issuer:** An optional component that a CA can delegate to publish CRLs.
- **Repository:** Any method for storing certificates and CRLs.

PKIX Management Functions

- **Registration:** A user first makes itself known to a CA
- **Initialization:** The client needs to be securely initialized with the public key and other assured information of the trusted CA(s).
- **Certification:** CA issuing a certificate
- **Key pair recovery:** Allows end entities to restore their encryption/decryption key pair from an authorized key backup facility.
- **Key pair update:** All key pairs need to be updated regularly
- **Revocation request:** An authorized person advises a CA of an abnormal situation requiring certificate revocation.
- **Cross certification:** Two CAs exchange information used in establishing a cross-certificate.

PKIX management protocols

- RFC 2510: Certificate Management Protocol (CMP)
- RFC 2797: Certificate Management Messages over CMS (CMC)

RFC2797 uses Cryptographic message Syntax CMS defined in RFC 2630