CSEN1001

Computer and Network Security

Mervat AbuElkheir
Ahmad Helmy
Mohamed Abdelrazik

Lecture (7)

Key Management II

Meet the seven people who hold the keys to worldwide internet security

It sounds like the stuff of science fiction: seven keys, held by individuals from all over the world, that together control security at the core of the web. The reality is rather closer to The Office than The Matrix

https://www.theguardian.com/technology/2014/feb/28/seven-people-keys-worldwide-internet-security-web

In a nondescript industrial estate in El Segundo, a boxy suburb in south- Los Angeles just a mile or two from LAX international airport, 20 people in a windowless canteen for a ceremony to begin. Outside, the sun is shi on an unseasonably warm February day; inside, the only light comes fro the glare of halogen bulbs.

ICANN's internet DNS security upgrade apparently goes off without a glitch

DNS Root KSK rollover happened Oct. 11 and will tighten security for the internet's address book

https://www.networkworld.com/article/3313341/icanns-internet-dns-security-upgrade-apparently-goes-off-without-a-glitch.html

By Michael Cooney

Network World | OCTOBER 12, 2018 08:20 AM PT

A few notes

1. RSA's proof of correctness
   - Euler: $a^{\phi(n)} mod\ n\ =\ 1$ where $\gcd(a,n) = 1$ (for $\gcd(M,n) = 1$)
   - Chinese Remainder Theorem (for $\gcd(M,n) \neq 1$)
2. Diffie-Hellman's secret key generation
   - $K_{AB} = a^{x_A x_B}\ mod\ q = {y_A}^{x_B}\ mod\ q = {y_B}^{x_A}\ mod\ q$
   - This will generate a secret key as a function of only two pieces of information by A and B. Even if a 3rd party M tries to generate a public/private key pair and generate K_{AM} it will not be equal to K_{AB} → $K_{AM} = a^{x_A x_M}\ mod\ q = {y_A}^{x_M}\ mod\ q = {y_M}^{x_A}\ mod\ q$
   - There is another way to include more than two parties $K_{ABM} = a^{x_A x_B x_M} mod\ q$

Applications of Public Key Algorithms

1. Digital Signatures

- Key Management and Distribution
  2. The distribution of symmetric keys (using symmetric keys!)
  3. The use of public-key encryption to create temporary keys for message encryption
  4. The use of public-key encryption to distribute secret keys
  5. The secure distribution of public keys

2- Mutual Authentication using the Needham-Schroeder Protocol

- Original third-party key distribution protocol
- for session between A, B mediated by KDC
- protocol overview is:

1. A→KDC: $ID_A \| ID_B \| N_1$

2. KDC → A: $E(K_a,[K_s||ID_B||N_1|| E(K_b,[K_s||ID_A])])$

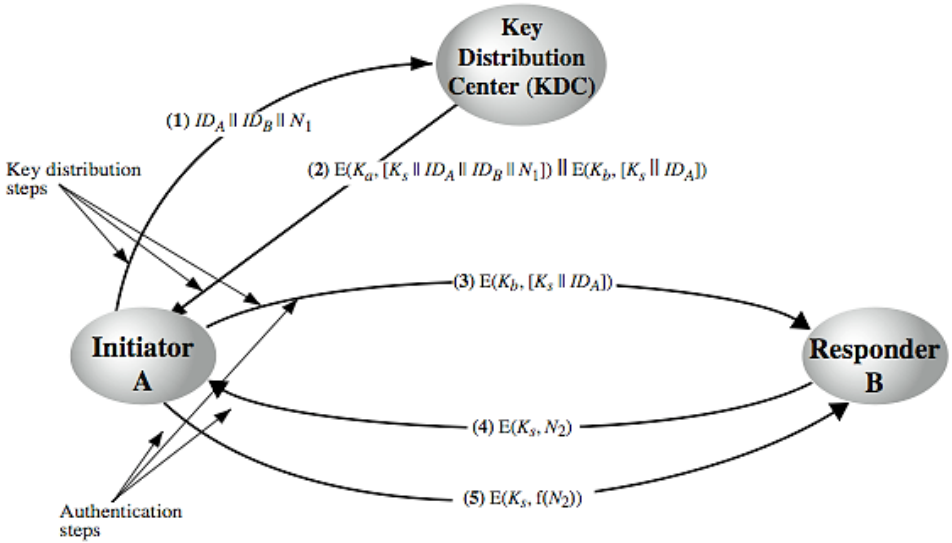
3. A → B: $E(K_b, [K_s||ID_A])$

4. B → A: $E(K_s, [N_2])$

5. A → B: $E(K_s, [f(N_2)])$

For replay attacks

Mutual Authentication using the Needham-Schroeder Protocol



Needham-Schroeder Protocol

- is vulnerable to a replay attack if an old session key has been compromised
  - then message 3 can be resent convincing B that it is communicating with A
- modifications to address this require:
  - timestamps in steps 2 & 3 (Denning 81)
  - using an extra nonce (Neuman 93)

Improvement using Timestamps

- protocol becomes (Denning):
  1. A→KDC: ID_A || ID_B
  2. KDC → A: $E(K_a,[K_s||ID_B||T$ || $E(K_b,[K_s||ID_A||T\ ])])$
  3. A → B: $E(K_b, [K_s||ID_A||T\ ])$
  4. B → A: $E(K_s, [N_1])$
  5. A → B: $E(K_s, [f(N_1)])$

1. A→KDC: ID_A || ID_B || N_1
2. KDC → A: $E(K_a,[K_s||ID_B||N_1||\ E(K_b,[K_s||ID_A])])$
3. A → B: $E(K_b, [K_s||ID_A])$
4. B → A: $E(K_s, [N_2])$
5. A → B: $E(K_s, [f(N_2)])$

- Success of this protocol relies on synchronized clocks at sender and receiver because both sides need to verify the timestamp
- Vulnerable to suppress-replay attack (when the sender's clock is ahead of recipient's clock)

Improvement using Nonces

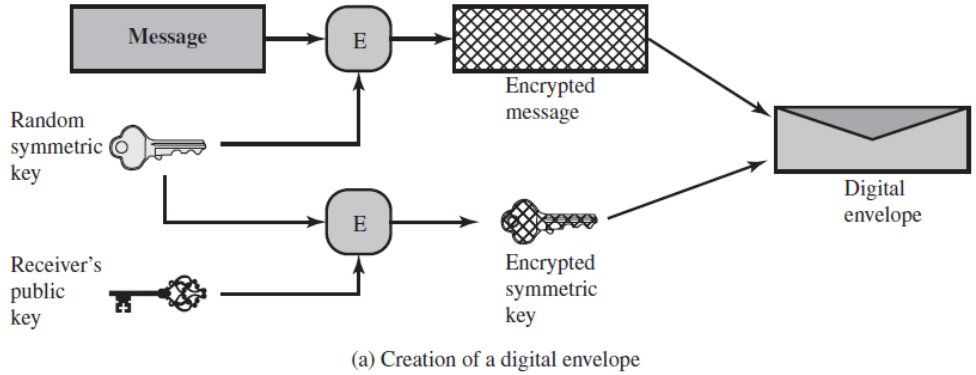
- protocol becomes (Neuman):
  1. A→B: $ID_A \,||\, N_A$
  2. B→ KDC: $ID_B||N_B||\ E(K_b,[ID_A||N_A||T_b])$
  3. KDC → A: $E(K_a, [ID_B||N_A||K_s||T_b])||E(K_b,[ID_A||K_s||T_b])||N_B$
  4. A → B: $E(K_b,[ID_A||K_s||T_b])||E(K_s, N_B)$

- Note that the timestamp is checked only at B → suppress-replay attack is not possible

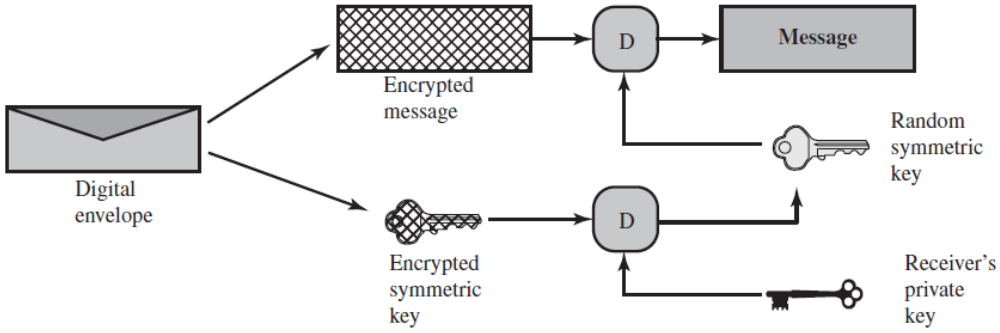
3- Digital Envelopes

Securely sending symmetric keys and encrypted message at once

Used in S/MIME
ElGamal is used for public-private key generation, and symmetric key is generated randomly

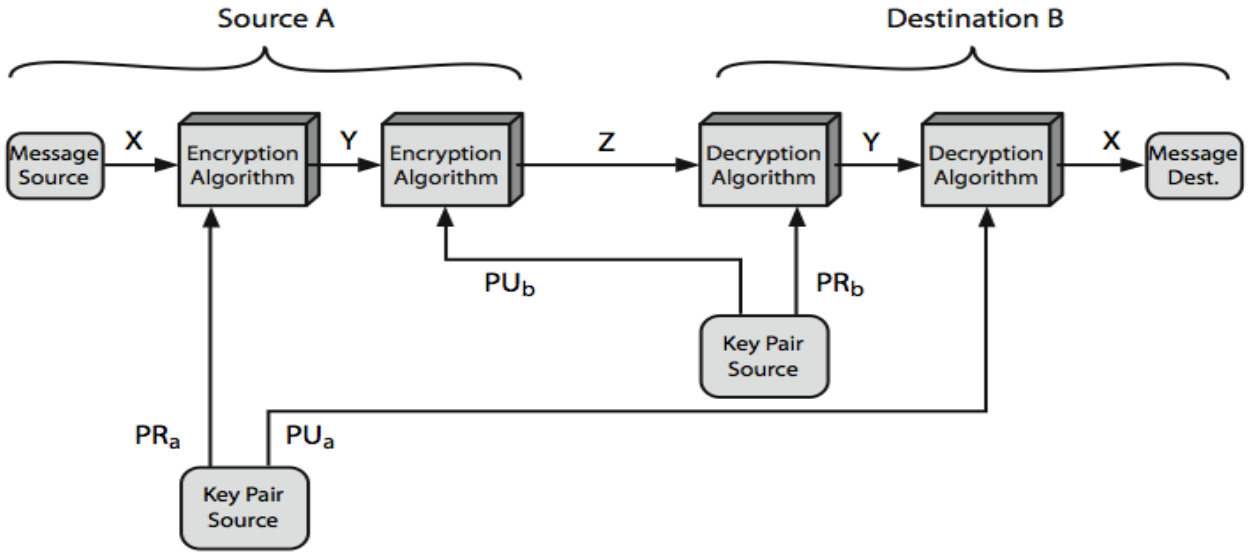


(a) Creation of a digital envelope



(b) Opening a digital envelope

NOTE: Digital envelopes are different from purely public key encryption/authentication!

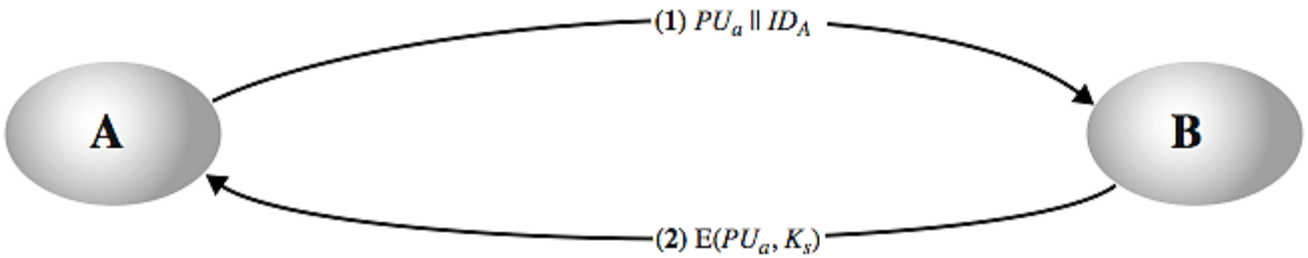


4- Secret Key Distribution Based on Public Key Cryptography

- Public key cryptosystems are often used to distribute secret keys
- Users could create random private/public D-H keys each time they communicate
- Users could create a known private/public D-H key and publish public in a directory, then others can consult and use to securely communicate with them
- Both scenarios are vulnerable to man-in-the-middle attacks → authentication of keys is needed!

Simple Secret Key Distribution Based on Public Key Cryptography

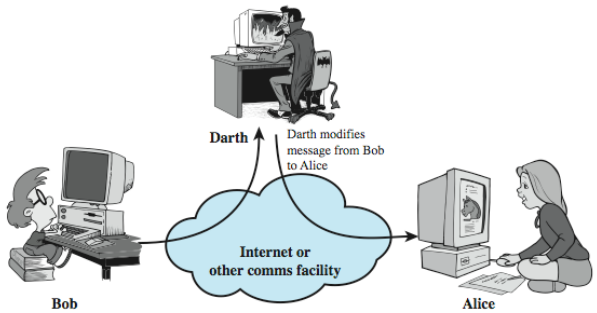
- Merkle proposed this very simple scheme
  - allows secure communications
  - no keys before/after exist



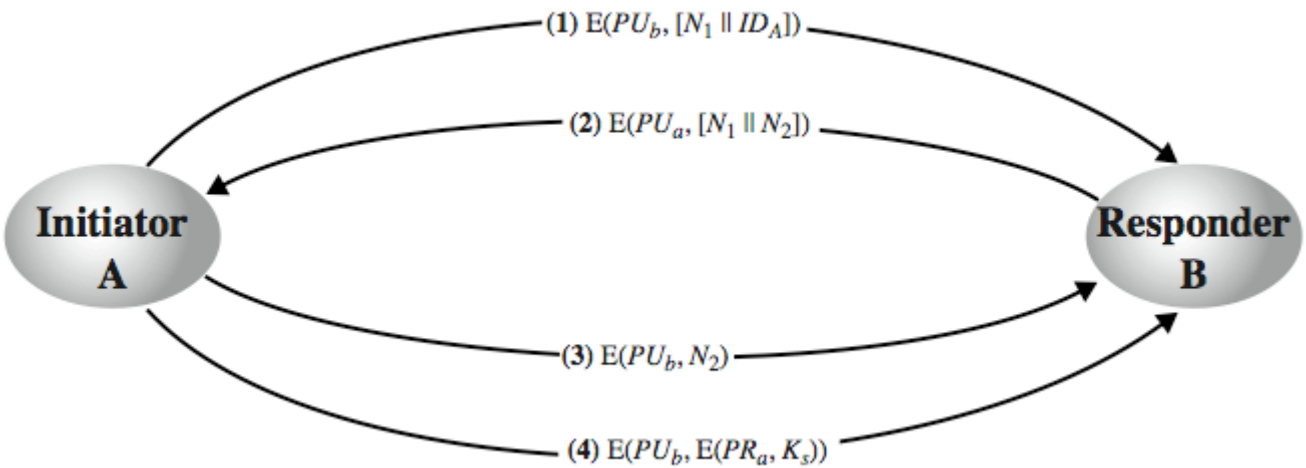
Vulnerability to MITM

An adversary, E, has control of the intervening communication channel, and performs the following

1. A generates a public/private key pair {PUa, PRa} and transmits PUa||IDa
2. E intercepts the message, creates its own public/private key pair {PUe, PRe} and transmits PUe || IDa to B
3. B generates a secret key, Ks, and transmits E(PUe, Ks)
4. E intercepts the message and learns Ks by computing D(PRe, E(PUe, Ks))
5. E transmits E(PUa, Ks) to A



Key Distribution with Confidentiality and Authentication



Hybrid Scheme
Session – Master - PK

- Retain use of private-key KDC
- Share secret master key with each user
- Distribute session key using master key
- Public-key used to distribute master keys
  - especially useful with widely distributed users
- Rationale
  - performance
  - backward compatibility

5- Distribution of Public Keys

- Can be considered as using one of:
  - Public announcement
  - Publicly available directory
  - Public-key authority
  - Public-key certificates

Public Announcement

- users distribute public keys to recipients or broadcast to community at large
  - e.g. append keys to email messages or post to news groups or email list
- major weakness is **forgery**
  - anyone can create a key claiming to be someone else and broadcast it
  - until forgery is discovered can masquerade as claimed user
  - We need to validate public keys!

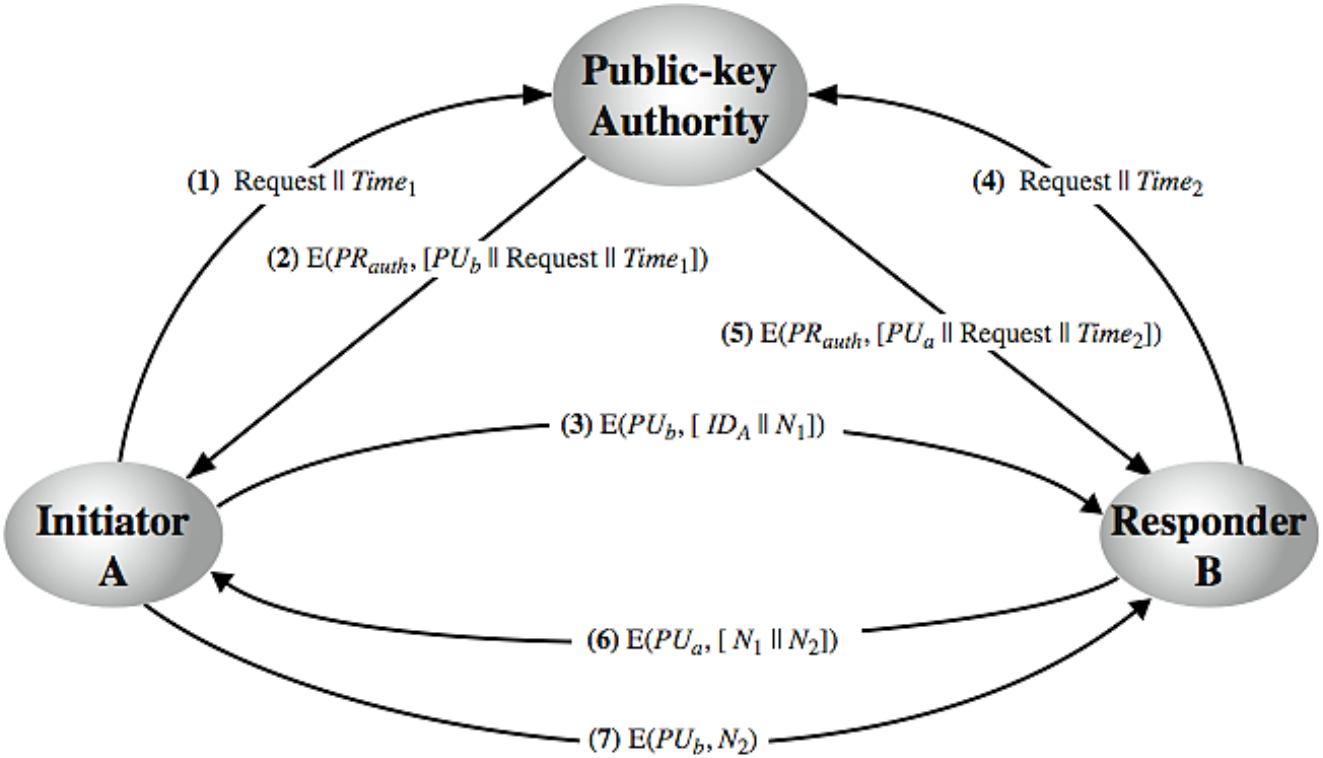
Publicly Available Directory

- can obtain greater security by registering keys with a public directory
- directory must be trusted with properties:
  - contains {name, public-key} entries
  - participants register securely with directory
  - participants can replace key at any time
  - directory is periodically published
  - directory can be accessed electronically
- still vulnerable to tampering or forgery

Publicly Key Authority

- improve security by tightening control over distribution of keys from directory
- has properties of directory
- and requires users to know public key for the directory
- then users interact with directory to obtain any desired public key securely
  - does require real-time access to directory when keys are needed

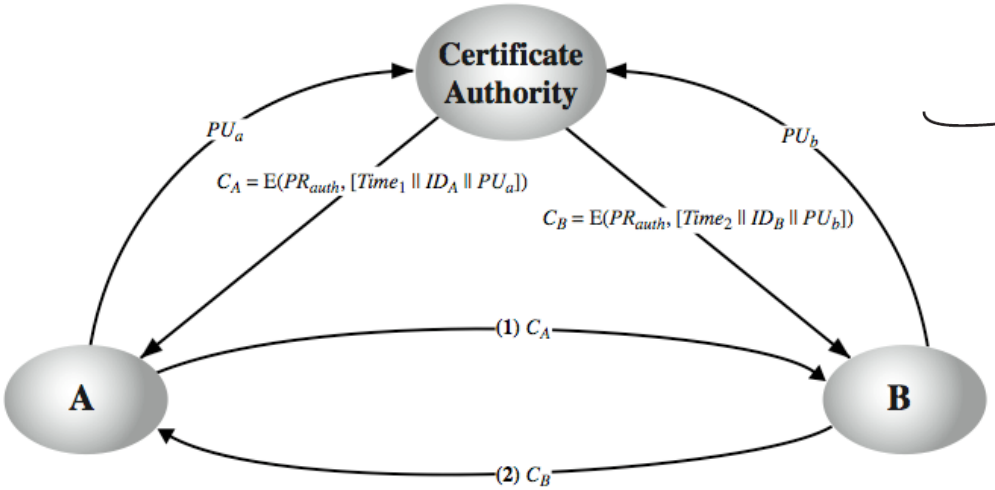
Publicly Key Authority

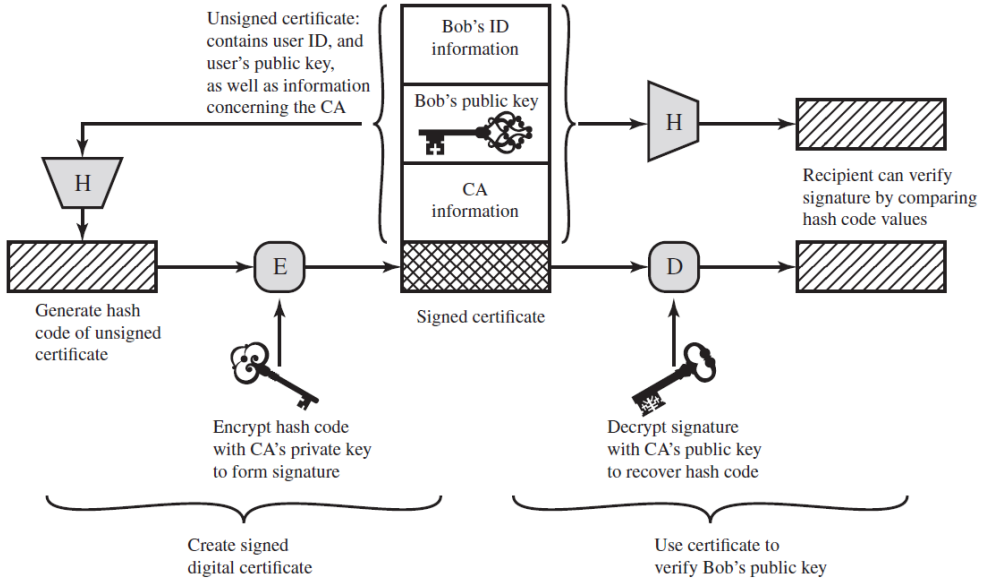


Publicly Key Certificates

- certificates allow key exchange without real-time access to public-key authority
- a certificate binds **identity** to **public key**
  - usually with other info such as period of validity, rights of use etc.
- with all contents **signed** by a trusted Public-Key or Certificate Authority (CA)
- can be verified by anyone who knows the public-key authorities public-key

Publicly Key Certificates





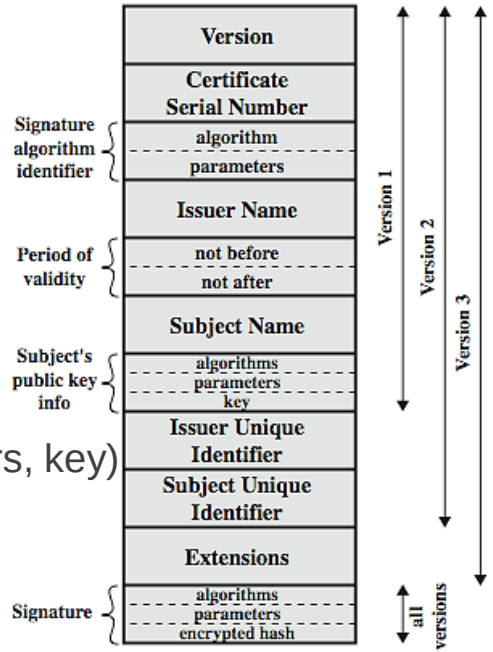
X.509 Certificate Use

X.509 Authentication Service

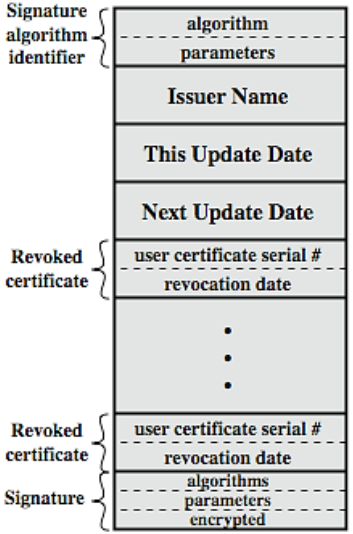
- part of CCITT X.500 directory service standards
  - distributed servers maintaining user info database
- defines framework for authentication services
  - directory may store public-key certificates
  - with public key of user signed by certification authority
- also defines authentication protocols
- uses public-key crypto & digital signatures
  - algorithms not standardised, but RSA recommended
- X.509 certificates are widely used
  - have 3 versions

X.509 Certificates

- issued by a Certification Authority (CA), containing:
  - version V (1, 2, or 3)
  - serial number SN (unique within CA) identifying certificate
  - signature algorithm identifier AI
  - issuer X.500 name CA
  - period of validity TA (from - to dates)
  - subject X.500 name A (name of owner)
  - subject public-key info Ap (algorithm, parameters, key)
  - issuer unique identifier (v2+)
  - subject unique identifier (v2+)
  - extension fields (v3)
  - signature (of hash of all fields in certificate)
- notation `CA<<A>>` denotes certificate for A signed by CA



(a) X.509 Certificate



(b) Certificate Revocation List

Obtaining Certificates

- any user with access to public key of CA can verify certificate
- only the CA can modify a certificate
- because cannot be forged, certificates can be placed in a public directory

CA Hierarchy

- if both users share a common CA then they are assumed to know its public key
- otherwise CA's must form a hierarchy
- use certificates linking members of hierarchy to validate other CA's
- each client trusts parents certificates
- enable verification of any certificate from one CA by users of all other CAs in hierarchy

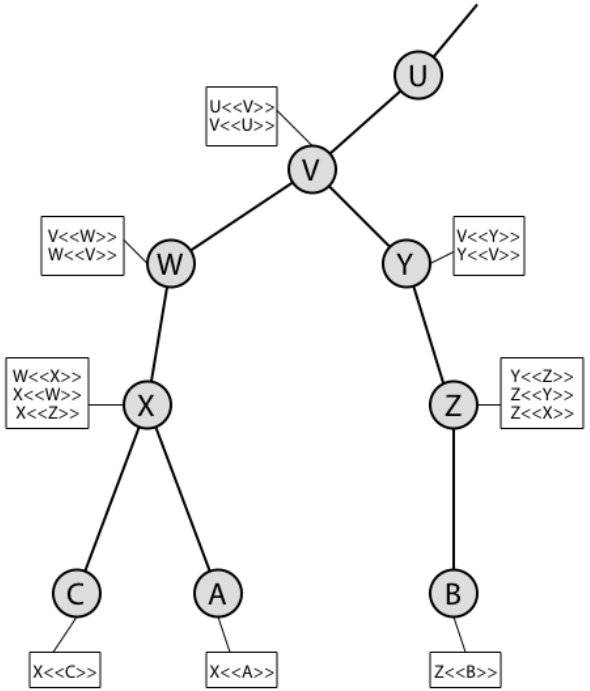
CA Hierarchy Use

- The directory entry for each CA includes two types of certificates:
  - **Forward certificates:** Certificates of X generated by other CAs, and
  - **Reverse certificates**: Certificates generated by X that are the certificates of other CAs

- In this example, we can track chains of certificates as follows:
  - A acquires B certificate using chain:

    X<<W>> W<<V>> V<<Y>> Y<<Z>> Z<<B>>
  - B acquires A certificate using chain:

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



Revocation of Certificates

- certificates have a period of validity
- may need to revoke before expiry, e.g.:
  1. user's private key is compromised
  2. user is no longer certified by this CA
  3. CA's certificate is compromised
- CA’s maintain list of revoked certificates
  - the Certificate Revocation List (CRL)
- users should check certificates with CA’s CRL

X.509 V.3

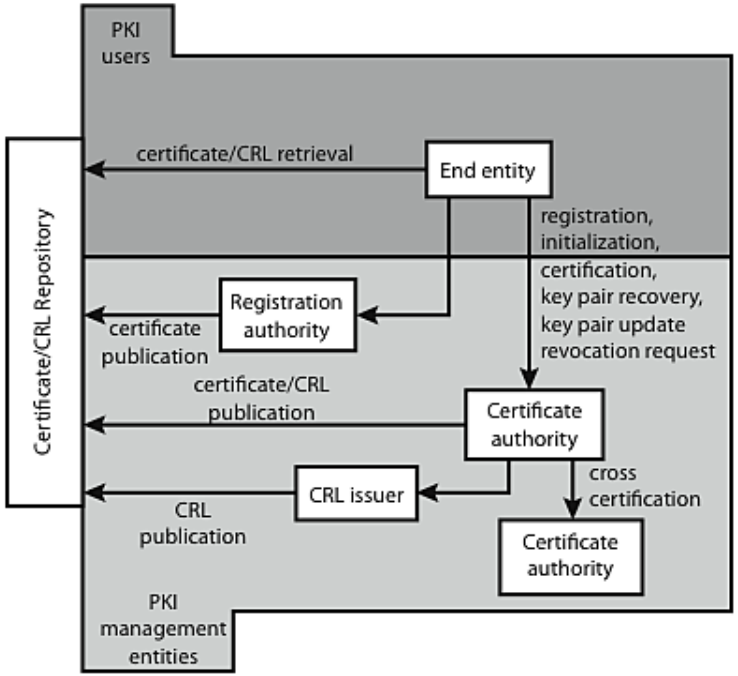
- has been recognised that additional information is needed in a certificate
  - email/URL, policy details, usage constraints
- rather than explicitly naming new fields defined a general extension method
- extensions consist of:
  - extension identifier
  - criticality indicator
  - extension value

Certificate Extensions

- key and policy information
  - convey info about subject & issuer keys, plus indicators of certificate policy
- certificate subject and issuer attributes
  - support alternative names, in alternative formats for certificate subject and/or issuer
- certificate path constraints
  - allow constraints on use of certificates by other CA's

Public Key Infrastructure

- End entity: A generic term used to denote end users, devices (e.g., servers, routers)
- Certification authority (CA): The issuer of certificates and (usually) certificate revocation lists (CRLs). It may also support a variety of administrative functions
- Registration authority (RA): An optional component that can assume a number of administrative functions from the CA. 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: A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by End Entities



PKIX Management

- **Registration:** whereby a user first makes itself known to a CA, prior to issue of a certificate(s) for that user. It usually involves some off-line or online procedure for mutual authentication
- **Initialization:** to install key materials that have the appropriate relationship with keys stored elsewhere in the infrastructure
- **Certification:** process where a CA issues a certificate for a user's public key, and returns it to the user's client system and/or posts it in a repository
- **Key pair recovery:** a mechanism to recover the necessary decryption keys when normal access to the keying material is no longer possible
- **Key pair update:** key pairs need to be updated and new certificates issued
- **Revocation request:** when authorized person advises need for certificate revocation, e.g. private key compromise, affiliation change, name change
- **Cross certification:** when two CAs exchange information used in establishing a cross-certificate, issued by one CA to another CA that contains a CA signature key used for issuing certificates