School of Electrical Engineering and Computing

SENG2250/6250
SYSTEM AND NETWORK SECURITY
(S2, 2020)

Key Management and Distribution

THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Outline

- Symmetric Key Management
  - *Key Transport*
  - *Key Distribution Centre*
- Public Key Management
  - *Key Agreement*
  - *Man-in-the-Middle (MITM)*
- Public Key Infrastructure

Key Management

- Generation
- Registration
- Distribution
- Storage
- Updating
- Deletion
- Backup/recovery
- Archiving

Key Generation

- Symmetric Key Cryptosystems
  - *Good Pseudo-Random or Random Number generator*
    - AES
    - DES
- Asymmetric key Cryptosystems
  - *Good Prime Numbers*
    - RSA
  - *Others*

Key Registration

- Associate Keys to entities
- Trusted Authorities
  - *E.g., Key Management Centre (KMC).*
  - *Symmetric System*
    - Secrecy and Integrity
  - *Asymmetric Systems*
    - Integrity of Keys

Key Management

- Storage
  - *Hosts*
  - *Network Components*
  - *Smart Cards*
  - *…*
- Distribution
  - *Secure Protocols*
  - *Initialization, Update, Deletion*
- Backup/recovery
- Archiving

Key Management

- Considerations of key management mechanism design
  - *What type of Keys?*
  - *Who generates Keys?*
  - *Where are they being generated?*
  - *Who needs the Keys?*
  - *How are they distributed?*
  - *How often Key changes required?*
  - *…*

Key Distribution

- Type of Keys
  - *Symmetric: for the system using symmetric key cryptography*
  - *Asymmetric: for systems using asymmetric (public) key cryptography*
- Symmetric Keys
  - *Remain as secret.*
  - *Distributed via secure channels.*
- Asymmetric Keys
  - *Public key is known to everyone, but private key remains secret.*
  - *Authenticity and integrity of public key is important. Why & How?*

Key Establishment

- Goal: Two users end up with a shared key that is only known by them.
- Possible ways of sharing a key:
  - *Two users use a supplementary secure channel, such as a courier service.*
    - **Disadvantages**: Costly, slow, questionable security.
  - *(Digital) Key exchange via a trusted authority (**T**):*
    - Each user can securely communicate with **T**, a central trusted authority.
    - **T** mediates between two users
    - **Disadvantages**: Requires a trusted node and creates a bottleneck. For every key between two users at least two communications involving **T** are necessary. **T** can be replaced by a network of authorities, but this increases the number of entry points for the intruder.
  - *Key exchange using public channels*

Key Establishment Protocols

- A **protocol** is an algorithm to achieve a certain goal:
  - *A sequence of steps precisely specifying the actions required of two or more parties .*
  - *Communicating parties in a protocol are called principals.*
- Key Establishment
  - *A process to make a shared secret key becomes available to two or more parties.*
- Key Transport
  - *Single party creates/obtains a secret and securely delivers to other(s).*
  - *One → Many*
- Key Agreement
  - *All parties cooperatively agree on a shared secret as a result of the protocol, using a function, say f.*
  - *No single party can predict/determine the shared key beforehand.*

Threats

- Disclosure
  - *Cannot access by unauthorised party.*
- Modification
  - *The integrity of message*
- Replay
  - *Is the message “fresh”?*
- Origin of Keys
  - *Is the key sent/agreed by the authentic user?*
  - *Impersonation?*

Key Transport Protocols

- Alice (A) and Bob (B) share a long-term key K_{ab}.
- $E(key; data)$ is a symmetric encryption scheme.
- K_s is a session key valid for a single secure communication between users.

A → B: $E(K_{ab};\ K_s)$

- Replay Attack
  - *Adversary can replay the previous message (key).*
  - *Reason: lack of key freshness.*
- Solutions
  - *Timestamp (TS)*

A → B: $E(K_{ab}; K_s, TS, B)$

  - *Time synchronisation issues.*

Challenge-Response

- Nonce
  - *To guarantee message freshness*
  - *Used only once*

- Challenge-Response Mechanism
  - *Uses Nonce*
  - *Authentication*

Example

- Challenge-response mechanism based solutions
  - *Key Transport Protocol*

    $B \rightarrow A\colon\ N_b$
    $\mathrm{A} \rightarrow B\colon\ E(K_{ab}; K_a, N_b, B)$
    Session key: K_a

  - *Key Agreement Protocol*

    $B \rightarrow A\colon\ N_b$
    $\mathrm{A} \rightarrow B\colon\ E(K_{ab}; K_a, N_b, N_a, B)$
    $\mathrm{B} \rightarrow A\colon\ E(K_{ab}; K_b, N_a, N_b, A)$
    Session key: $f(K_a, K_b)$

  - *Communication overhead in which requires more than one message. Efficiency* ***vs.*** *Security*

Key Establishment with a Server

- Trusted Third Party (TTP)
  - *An authority trusted by all users.*
- Key Distribution
  - *Key Distribution Centre (KDC): supplies the session key.*
  - *Key Translation Centre (KTC): enables a session key chosen by one user to be available to others.*
- A and B have shared long-term keys with a TTP, e.g, KDC and KTC.

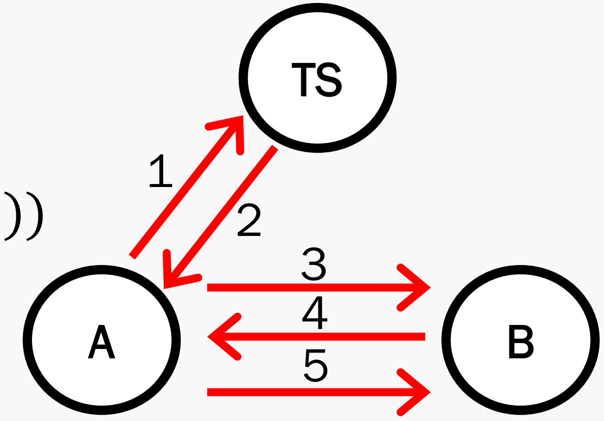
Needham-Schroeder Protocol

- Entities
  - *TS is a trusted server.*
  - *A and B are users who have shared long-term keys with TS.*
- Role of TS
  - *Key distribution centre*
  - *Generate session keys*
- The NS protocol is the basis of many server based key distribution systems, including Kerberos. We will introduce it later.

Needham-Schroeder Protocol

- E is a symmetric encryption scheme.
- N_a and N_b are nonce chosen by A and B, respectively.
- K_{as} and K_{bs} are shared long-term keys between A (respectively, B) and TS.
- K is a session for communication between A and B.
- The Protocol

1. $A \rightarrow TS: A, B, N_a$
2. $TS \rightarrow A: E(K_{as}; K, B, N_a, E(K_{bs}; K, A))$
3. $A \rightarrow B:\ E(K_{bs}; K, A)$
4. $B \rightarrow A:\ E(K; N_b)$
5. $A \rightarrow B:\ E(K; N_b - 1)$



PKC Based Key Transport

- One-pass key transport:

  $A \rightarrow B: E(PK_B; k)$

  - *E is a public key encryption system.*
  - *PK_B is Bob's public key.*
  - *k is a key chosen by Alice.*
- Issues
  - *A will not know if B receives the message.*
  - *B does not know the source of the message.*
  - *Replay attack*
- Solution

  $A \rightarrow B: E(PK_B; k, IV, \text{Alice's id}, \text{timestamp}, \text{seq number})$

  - Secure against Replay Attack.
  - How to design a protocol using challenge-response mechanism?

Preliminaries - Group

- Group is a mathematical representation of symmetries.
- A set of elements and a binary operation + ("addition"/composition) "on" that set, satisfying the following properties.
  - *(A1)* ***Closure***: $(a + b) \in \mathbb{G},\ \forall a, b \in \mathbb{G}$
  - *(A2)* ***Associativity***: $a + (b + c) = (a + b) + c,\ \forall a, b, c \in \mathbb{G}$
  - *(A3)* ***Identity***: $\exists \mathrm{e} \in \mathbb{G}: a + e = e + a = a,\ \forall a \in \mathbb{G}$
  - *(A4)* ***Inverse***: $(\forall a \in \mathbb{G}), \exists (a' \in \mathbb{G}): a + a' = a' + a = e$

Preliminaries - Abelian Groups

- A *group* provides commutative.
  - *(A5)* ***Commutative:*** $a + b = b + a, \forall a, b \in \mathbb{G}$
- The set of integers $\{-5, -4\ , 0, 4{,}5\}$ under integer addition IS an Abelian group.
  - *What is the identity?*
  - *What is the inverse of each element?*
  - *Why is it Abelian?*

Preliminaries - Generator

- We define exponentiation within a group by repeated application of the operation +.
  - $g^0 = e,\ g^1 = g,\ g^2 = g + g,\ g^{i+1} = g^i + g,\ \ldots$
  - *$g \in \mathbb{G}$ is an element of group $\mathbb{G}$*
  - *e is the identity of $\mathbb{G}$.*
- If every element of $\mathbb{G}$ can be written in the form of g^i for some $g \in \mathbb{G}$, then g is a **generator** of $\mathbb{G}$ and the group is said to be cyclic.
- A generator is also known as a **primitive root**.

Discrete Logarithm Problem (DLP)

- Definition
  - *Let g be a generator of multiplicative group $\mathbb{G}$, given $h \in \mathbb{G}$, find $a \in \mathbb{Z}_p^*$, such that $h = g^a \bmod p$.*
  - *p is a large prime.*
- It is computationally difficult to find:
  - a
  - *The multiplicative inverse (h^{-1}) of h, s.t $h^{-1} = g^{-a}$ mod p*
- It provides a way to set up public and private key pairs.

Key Transport with Public Key Cryptography

- Shamir's no-key protocol

$\mathrm{A} \rightarrow B\colon\ K^{a}\ mod\ p$
$\mathrm{B} \rightarrow A\colon\ (K^{a})^{b}\ \ mod\ p$
$\mathrm{A} \rightarrow B\colon\ (K^{ab})^{a^{-1}}\ \ mod\ p$
Session key: K

1. *Select & publish a prime p such that for that prime Discrete Logarithm (DL) problem is hard.*
2. *A does the following steps*
    1. Selects a random number a , $1 < a < p-2$, a co-prime to $p-1$.
    2. Computes a^{-1} mod p−1.
3. *B does the same as A and obtains b and b^{-1} mod p-1.*

Example

- System Parameters
  - $p = 19, K = 3$
  - $a = 5,\ a^{-1} = 11$, i.e $a^{-1}a = 1\ mod\ 18$
  - $b = 7,\ b^{-1} = 13$, i.e $b^{-1}b = 1\ mod\ 18$
- Shamir's no-key protocol
  - *Session key: $K =$ 3, chosen by A.*

    $\mathrm{A} \rightarrow B\text{:}\ 3^5\ mod\ 19 = 15$
    $\mathrm{B} \rightarrow A\text{:}\ (3^5)^7\ \ mod\ 19 = 13$
    $\mathrm{A} \rightarrow B\text{:}\ (3^{5\times 7})^{11}\ \ mod\ 19 = 2$

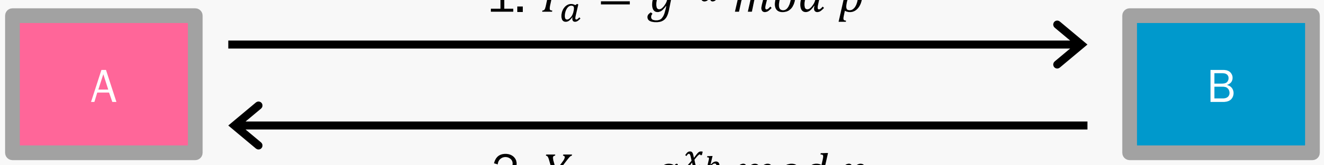
  - *Session key: $2^{13}\ mod\ 19 = 3$, calculated by B.*

Diffie-Hellman Key Agreement

- System Setup
  - *A and B agree on a common prime p and a generator g of $\mathbb{Z}_p^*$.*
  - ***Safe prime*** *: $p = 2q + 1$, where q is a prime.*
- Private Keys
  - *A chooses a random number x_a, $1 < x_a < p$, as a private key*
  - *B chooses a random number x_b, $1 < x_b < p$, as a private key.*
  - *These are usually used as ephemeral keys of a session, i.e choose different keys at each protocol execution.*
- Public Keys
  - *A computes the public key $Y_a = g^{x_a} \bmod p$.*
  - *B computes the public key $Y_b = g^{x_b} \bmod p$.*

Diffie-Hellman Key Agreement

1. $Y_a = g^{x_a} \, mod \, p$



2. $Y_b = g^{x_b} \, mod \, p$

3. $K_{ab} = Y_b^{x_a} = g^{x_b x_a} \, mod \, p$

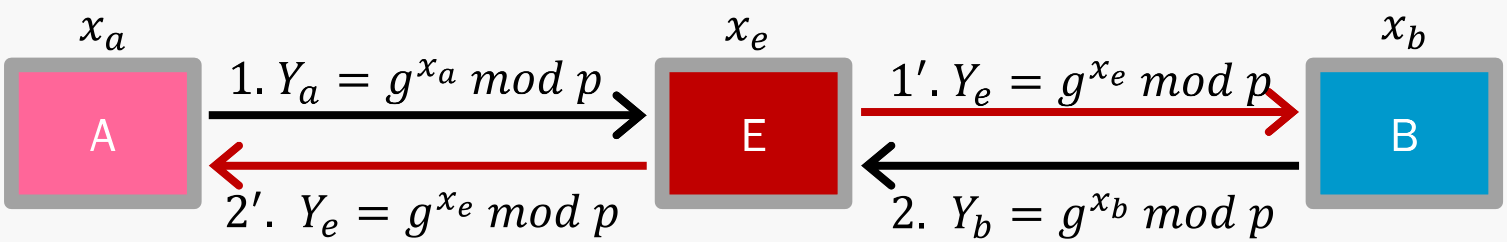
3. $K_{ba} = Y_a^{x_b} = g^{x_a x_b} \, mod \, p$

Session key: K_{ab}=K_{ba}

Example

- System Setup
  - $p = 19$
  - $g = 2$
- Private Keys
  - $x_a = 3$
  - $x_b = 5$
- Public Keys
  - $Y_a = g^{x_a} = 2^3 \ mod \ 19 = 8.$
  - $Y_b = g^{x_b} = 2^5 \ mod \ 19 = 13.$
- Session
  - $K_{ab} = Y_b^{x_a} = 13^3 \ mod \ 19 = 12.$
  - $K_{ba} = Y_a^{x_b} = 8^5 \ mod \ 19 = 12.$

Man-in-the-Middle (MITM) Attack



3. $K_{ab} = Y_e^{x_a} = g^{x_e x_a} \mod p$

3. $K_{ba} = Y_e^{x_b} = g^{x_e x_b} \mod p$

Session key: $K_{ab} \neq K_{ba}$

- Adversary E modifies messages without being detected.
- Consequently, E shares session keys with A and B, respectively.
- A and B thought they are communicating with each other, which is not.
- No security. ☹
- How to solve the problems? (Hint: add key authenticity check, e.g., public-key certificate)

Forward Secrecy

- A key agreement protocol provides forward secrecy, if old session keys remain secure when the long-term keys of participated entities were compromised.
- Also known as *perfect forward secrecy*.
- Does NS protocol provide forward secrecy?

More Concepts in Key Agreement Protocols

- Key Confirmation
  - *One party is assured that all other parties actually have possession of particular secret key.*
- Implicit Key Authentication
  - *One party is assured that only specifically identified parties can derive a particular key.*
- Explicit Key Authentication
  - *key confirmation + implicit key authentication*

Remarks of Protocols

- Why did we review “old” and “flawed” protocols?
  - *They are well-known and simple to show challenge-response mechanisms.*
  - *The secure protocols are usually developed from “old and flawed” versions.*
  - *By learning the attacks (flaws) and their solutions, it gives us lessons for new protocol design.*
  - *DH protocol is one of the most important key exchange protocol used in practice.*

Public Key Management

- Generation of Public Keys
- Storage of Public Keys
- Integrity of Public Keys
- Masquerading of Users
- Trusted Authority

Key Authentication

- A public key should be authenticated in some way and it can be (publicly) verified.
- Authentication of public keys binds identity of key owner and the public key.
    - *Certificate*
- Public Key Infrastructure (PKI) provides a mechanism.

Public Key Infrastructure (PKI)

- Defined by **RFC2822**
- PKI
  - *Set of hardware, software, people, policies and procedures needed to create, manage, store, distribute, and revoke public key Certificates.*
- Objective
  - *Enable secure, convenient and efficient acquisition of public keys.*
- Related Standards
  - *E.g, X.509, PKIX, SPKI.*

Public Key Infrastructure X.509 (PKIX)

- PKIX is a formal and generic model based on X.509. (Driving force by IETF PKIX working group)
- Provides a set of management functions (APIs).
- PKIX Architecture Model Components
  - *Certificate Authority (CA)*
  - *Registration Authority (RA)*
  - *End entity: Users who are certified by CAs*
  - *Repositories: Stores certificate and CRLs*
  - *CRL issuer: Optional component to publish CRLs*

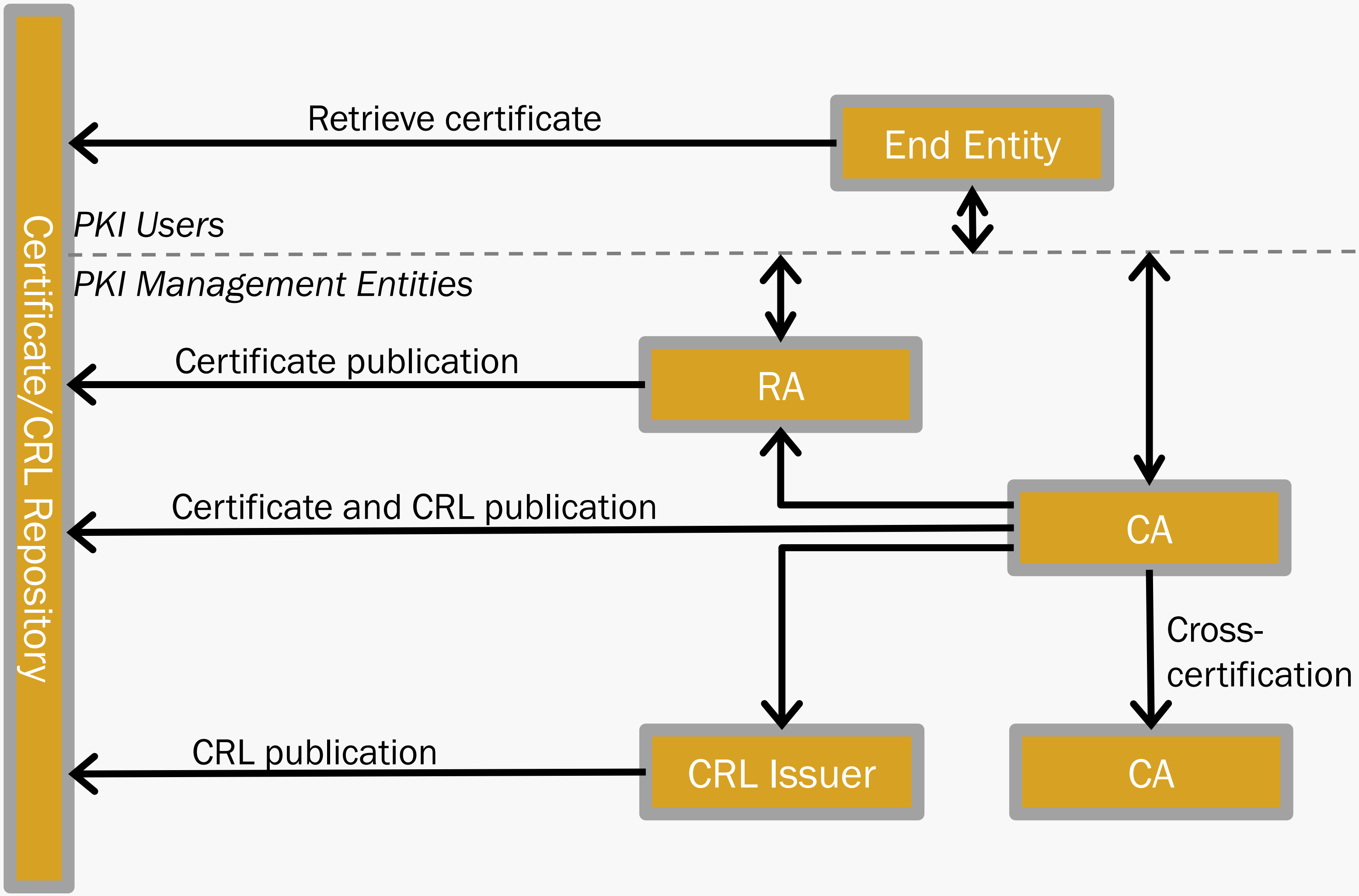
PKIX Management Functions

- Registration
  - *Verify user information.*
- Initialisation
  - *Get information to start communication with PKI.*
- Certification
  - *CA issues certificate for user’s public key.*
- Key Pair Recovery
  - *Allows user to recover the private key.*

PKIX Management Functions

- Key Pair Update
  - *Allows user to update the key pair.*
- Revocation Request
  - *Authorised user can request to revoke particular certificate.*
- Cross Certification
  - *One CA issues certificate for another CA.*

PKIX Architecture Model



Certificate Authority

- Trusted by subscribers.
- Maintains and issues certificate.
- Identifies entity whose Certificate Signing Request (CSR) is submitted.
- Responsible for the security of certification process.
- Has public and private key pair.

Certificate

- Certificate is a publicly verifiable record which associates a user and a public key.
- Certificate is generated by a trusted authority.
  - *CA*
  - *Digital signature of CA*
- Certificate verification
  - *Offline: based on the static information, e.g, known verification key and text description.*
  - *Online: dynamically checking the validity of certificate with servers, i.e, trusted authority.*

X.509 Certificate V3

Version	Certificate format: v1, v2, v3
Certificate Serial Number	Identifier of the certificate within CA
Signature Algorithm Identifier: *Algorithm and parameters*	Signature algorithm and parameters used to sign the certificate
Issuer Name	Name of CA
Period of Validity: *Not before and not after date*	The lifetime of certificate
Subject Name	Name of user, i.e key owner
Public Key Information: *Algorithm, parameters and key*	Public key of subject associated with algorithms and parameters to be used in.
Issuer Unique Identifier	(v2) Unique identifier of CA
Subject Unique Identifier	(v2) Unique identifier of subject
Extensions	(v3) extension fields
Signature: Algorithm, parameters and encrypted hash	Generated by CA that contains hash code of other fields, signature algorithm, etc.

X.509 Certificate V3 Extensions

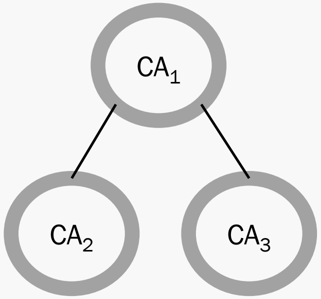
- Authority Key Identifier
  - *Issuer has multiple signing keys*
  - *Identification : Issuer Certificate, Name, Serial No.*
- Key Usage
  - *The purpose of the key defined in the Certificate*
  - *Key used for verifying signatures*
  - *Key used for encryption*
- Policy Information
  - *Policy under which the Certificate is issued and the purposes for which the Certificate can be used.*

X.509 Certificate

- Certificate Generation
  - *Check subject information*
  - *Compute digest of subject information using specified algorithms*
  - *Sign the digest using CA's private key*
  - *Fill in fields of X.509 certificate.*
- Certificate Verification
  - *Verify validity of CA's signature using the specified algorithms.*
  - *Check validity of certificate.*
- Need PKI support!!

X.509 Hierarchy

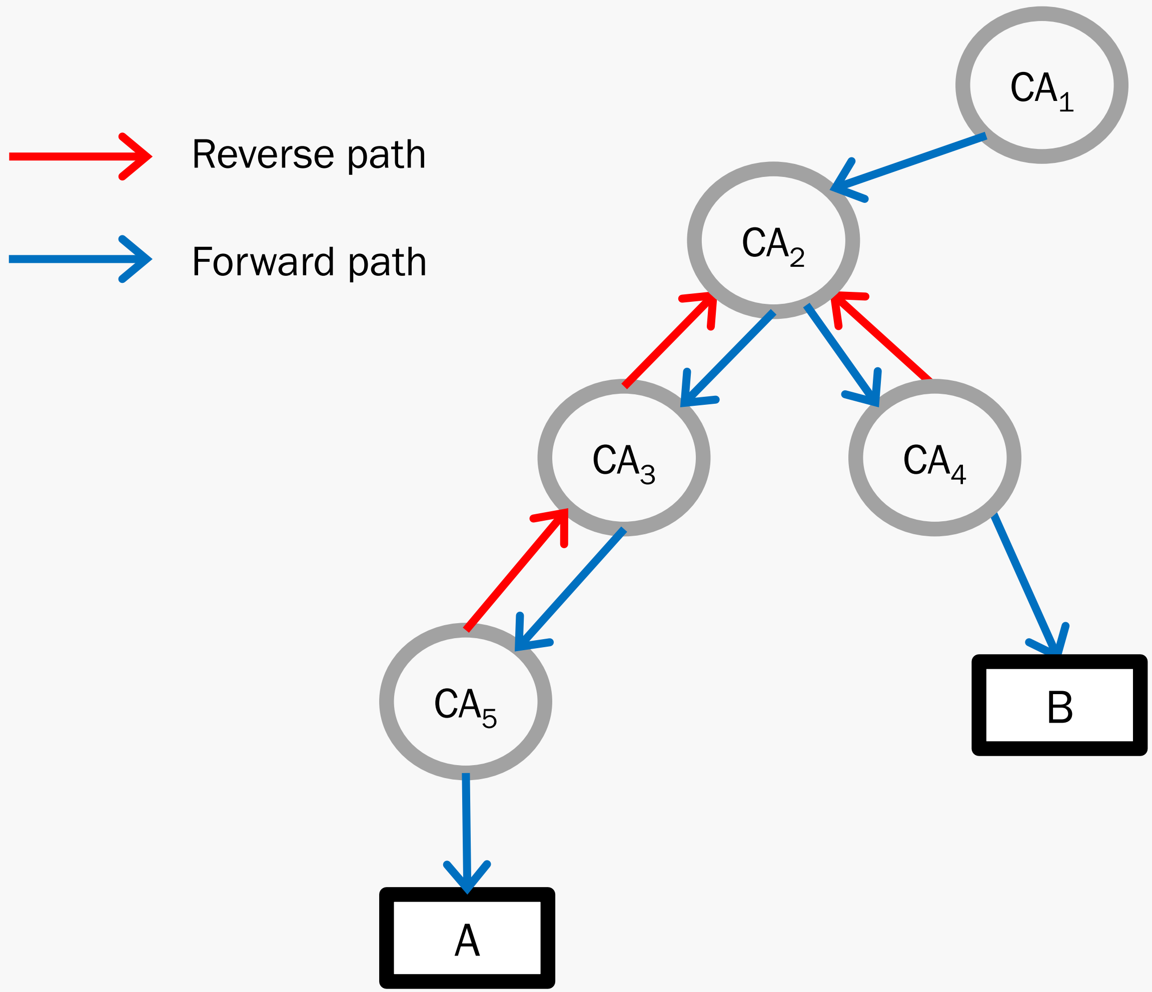
- Directory
  - *Online.*
  - *Stores certificates.*
  - *Publicly accessible.*
- Multiple CAs
  - *Directory Information Tree (DIT).*
  - *Find common ancestor CA to verify certificate.*



X.509 Hierarchy

- Common CA
  - *Shared by users.*
  - *Use common CA's public key to start certificate(s) verification.*
- Certification Path
  - *Users does NOT share a common CA.*
  - *Certificate Chain.*
  - *Subject certificate ←→ target certificate.*
  - *Contains all intermediate certificate needed.*
  - *Each CA has certificate for its parent and child CAs.*

X.509 Hierarchy



THE UNIVERSITY OF NEWCASTLE AUSTRALIA

Example

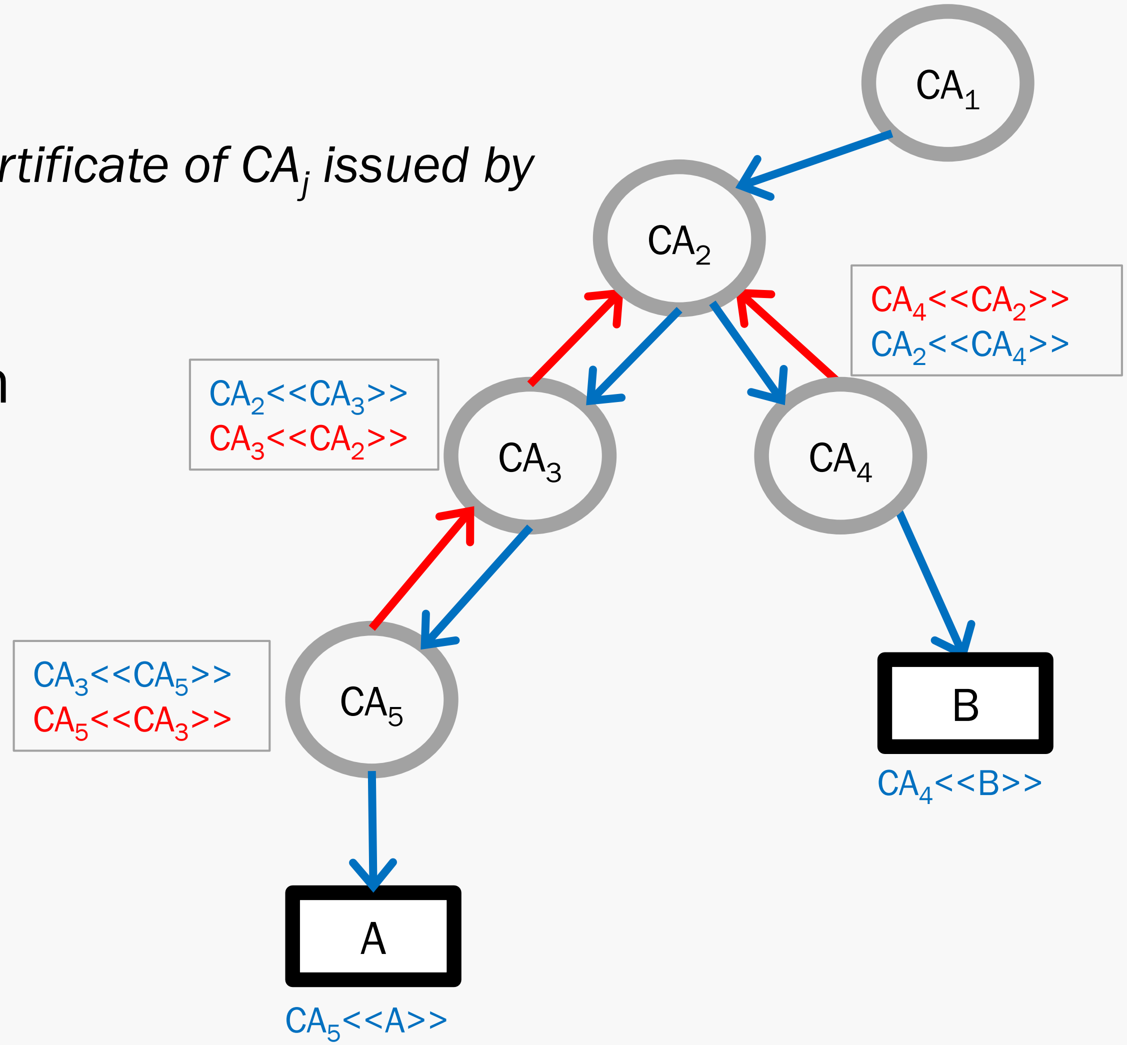
- Notations
  - *$CA_i<<CA_j>>$: certificate of CA_j issued by CA_i*
- Certification Path
  - *$A \rightarrow B$:*

$CA_5<<CA_3>>$

$CA_3<<CA_2>>$

$CA_2<<CA_4>>$

$CA_4<<B>>$



Self-Signed Certificates

- User signed certificate
  - *No CA*
- Root CA
  - *Authority trusted by other subscribers.*
  - *Self-sign itself.*
- Trust Level
  - *User signed: very limited trust on the certificate.*
  - *Root CA signed: widely trusted.*
- MIMT
  - *User signed certificate has NO protection against MITM attacks.*

Certificate Revocation

- Certificate Lifetime
  - *Not Before Date*
  - *Note After Date*
- Revoke in case, e.g.,
  - *User's private key was compromised.*
  - *CA's certificate or private key was compromised.*
  - *User is no longer being certified with the CA.*
- Methods
  - *Certificate Revocation List (CRL)*
  - *Online Certificate Status Protocol (OCSP)*

Certificate Revocation List

- CRLs must be periodically updated.
- Check CRLs during certificate verification
  - *May not be configured by default.*

Signature Algorithm Identifier: *Algorithm and parameters*
Issuer Name
This Update Date
Next Update Date
Revoked Certificate 1 *Certificate serial number and revocation date* … **Revoked Certificate n** *Certificate serial number and revocation date*
Signature: Algorithm, parameters and encrypted hash

Registration Authority (RA)

- Assess relationship between user identity and public key.
  - *Verify user request for certificate*
- Optional
  - *Can be co-implemented with CA.*
- User must prove he/she know the private key related to the public key which is being certified.