

## Topic 2. Security Infrastructure

- **Infrastructure** Support
- **Authentication** Server
  - ▶ **Entity** Authentication
  - ▶ Authentication Server
- **Certificate** Authority
  - ▶ **Public Key** Certificate
  - ▶ Certificate Chain and **Revocation**
- Key **Generation** and Distribution Server
  - ▶ Public/Private Key Pair Generation
  - ▶ Key **Escrow**
  - ▶ Symmetric Key Generation and Distribution
- **Signing** Server
  - ▶ Signature for Authorized Software
  - ▶ Signature for Copyrights

# 1. Infrastructure Support

- A security mechanism consists of **protocols** and algorithms to establish certain trust relationships in a communication system.
- **infrastructure support** is a set of services to establish trust relationships.
- Examples:
  - ▶ an enterprise IT department may **generate** and **distribute keys** for the wireless devices used within the enterprise;
  - ▶ **certificate** authorities.

## 2. Authentication Server

### 2.1 Entity Authentication

- Entity authentication is a process for one party to **verify** another party's identity with whom it communicates.
- Two types of an entity authentication: it may be
  - ▶ a user authentication or
  - ▶ a device authentication.
- The verification is to check whether the **specific** authentication data is valid, which can be generated **only** by the party with the claimed identity.
- The party to be **authenticated** must provide some verifiable evidence to prove it is the entity identified by the identity.

# Methods for Entity Authentication

- **Password** based (classic method).
- Cryptography based authentication: **challenge-response method**, which can be either public key based or symmetric key based.
  - ▶ Symmetric key based challenge-response method: assume that both entities  $I$  and  $J$  share a key  $K_{IJ}$ .

## Challenge-response method by MAC

**MAC** is a message authentication code. Assume that **party J** wishes to authenticate party *I*.

- **Party J**: generates a random string *Ch*, called a **challenge**, and send to party *I*.
- **Party I**: generates a **response** *Res* by computing

$$Res = MAC(K_{IJ}, Ch)$$

and sends *Res* to party *J*.

- **Party J**: calculates

$$Res' = MAC(K_{IJ}, Ch)$$

and checks

$$Res = Res'$$

If it is true, then party *J* can be insured that it is party *I*, since **no one else** holds key  $K_{IJ}$  and can calculate a correct *Res*.

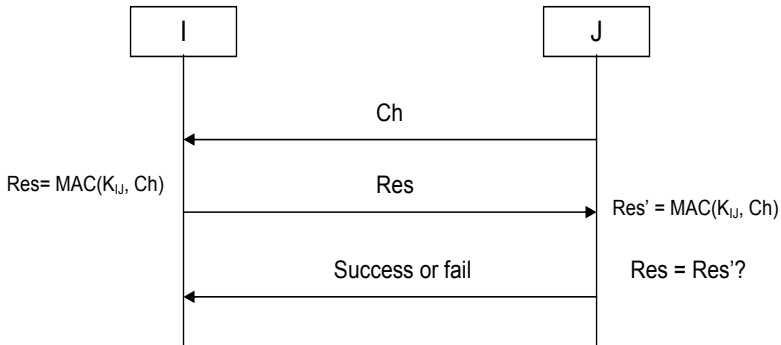


Figure: Entity Authentication by Challenge-Response

# Functionalities of Authentication Server

- Entity authentication can be executed as an access authentication.
- The network entity which authorizes access is called ***a network access server***:
  - ▶ In cellular network, an **access server** is the entity which authorizes the access.
  - ▶ In IEEE 802.11 network (wireless local area network), **an access point** plays the role of a network access server.

# Centralized Model

- An authentication server is shared by many **access servers**. Not like each access server, an authentication server may be located remotely.

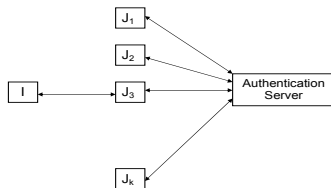


Figure: Authentication Server

The figure illustrates a situation that an entity  $I$  may **request access** from any of the access servers  $J_1, J_2, \dots, J_k$ , while each access server conducts authentication through the authentication server.

### 3. Certificate Authority

**Question:** Why public-keys cannot be directly distributed in an open channel? How can Alice be assured that the key is Bob's but not Eve's public key?

- The **above concern** arises from the fact that in a public key cryptography scheme, it is feasible for any party to generate a pair of public and private keys.
- Eve can generate such a key pair and to **fool others** by claiming that it is Bob's public key.
- The information sent to Bob will be **encrypted** by the public key Eve generated, for which Eve knows the corresponding private key.
- Thus, **instead of Bob**, Eve can decrypt the information supposed to be accessed only by Bob.

**A public key must binding with its owner in a trusted way!**  
**Certificate authority is needed!**

## 3.1. Public Key Certificate

For a given public key and its owner,

**a certificate is a binding between the public key and the owner's identity**

**Method** for insuring such a binding: digital signature.

The CA will **sign the public key together with its owner's identity** in order to generate a verifiable signature

**Format:**

**Certificate** of Bob's public-key  
=  $\text{Signature}_{CA}(\text{Bob's public key,}$   
**Bob's identity,**  
the expiration date,  
**the scope (enc or sign))**

# How to verify the validity of CA's public-key?

Since a **certificate** can be verified by CA's public key, it seems we are back to the original question, that is, how it can be insured that it is CA's but not Eve's public key.

A CA's public key must be able to be **recognized as authentic**. Otherwise, it has no help by introducing a CA.

Introducing a CA **converts a peer-to-peer trust** assumption to a **multiple-to-one trust** assumption.

## 3.2. Certificate Chain and Revocation

- A system may consist of **multiple administrations**. Each administration has its own certificate authority,

$$CA_1, CA_2, \dots, CA_n,$$

respectively.

- Any two entities,  $I$  and  $J$ , belonging to two different administrations cannot establish trust unless two **administrations have a commonly trusted third party**.
- Solution: **a certificate hierarchy**, i.e., it is needed a root CA trusted by different administrations.

# Certificate Hierarchy

A higher level certificate authority will sign certificates of the lower level certificate authorities.

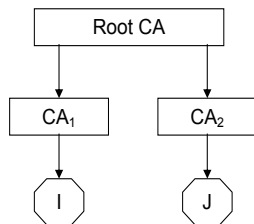


Figure: Certificate Hierarchy

**The root CA will sign certificates** for  $CA_1$  and  $CA_2$ .

Then,  $CA_1$  and  $CA_2$  will **sign certificates** of the entities belonging to its own administration.

# Verification of Certificate hierarchy

If entity  $I$  need to verify a public key certificate of the entity  $J$ , logically, the verification will include the following two steps.

- 1 **Verify the certificate of  $CA_2$ 's public key.** That is, verify the signature of root CA with root CA's public key. If it is valid, then go to step 2.
- 2 **Verify the certificate of  $J$ 's public key.** That is, verify the signature of  $CA_2$  with  $CA_2$ 's public key. If it is valid, then  $J$ 's public key is valid.

# Certificate Chain

This trust relationship can be described as a trust chain. We also call it *a certificate chain*.

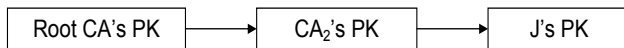


Figure: Certificate Chain

# Revocation

- A certificate may have to be revoked before its expiration date.
- **Certificate revocation list (CRL)**: For certificate verification, it may have to verify whether it is included in a certificate revocation list (CRL). Certificate authority is the entity to provide a trustable and updated CRL.

## 4. Key Generation and Distribution Server

### 4.1. Public/Private Key Pair Generation

**Difficulties to generate public/private key pairs: complicated mathematics operations.**

**A third part is needed to generate public/private key pair**, e.g., a certificate authority. **In this case, the private key will need to be transported to the owner in a protected manner.**

Whether employing a third party to generate key pairs is a decision made on multiple factors. However, if a third party is used, then it must facilitate a secure key distribution system.

## 4.2. Key Escrow

- **A key escrow server is used for a service to provide back up** for encrypted contents.
- It is often a **key generator**. Sometimes, a digital content provider will escrow the keys to retrieve the protected digital products in case an eligible customer lost its key.
- A key generator may not be a key escrow server. For a key generator, it will not provide any back up service.
- **For identity based encryption (IBE)**, in order to imbed an identity to a public key, it relies on a trusted third party to generate the **corresponding private key**.

## 4.3. Symmetric Key Generation and Distribution

A key should be generated by a sound **pseudo-random sequence/number generator**.

### Public-key case

A symmetric key may be generated and shared among different parties through a **public key method** without depending on a key generation and distribution server.

### Symmetric key case

(1) Key distribution server is used in a system may only use symmetric key based cryptography mechanisms.

(2) Methods for key distribution: a) accomplished in the manufacture by provisioning the keys to each device; b) distributed through service providers.

(3) If the keys must be distributed in an unprotected environment, then it **requires a pre-established cryptographically protected** channel between each entity and the key distribution server.