

## Chapter 2. Security Infrastructure

Generally speaking, a security mechanism consists of protocols and algorithms to establish certain trust relationships in a communication system. However, the trust relationship cannot be established from the air. It depends on already established relationships to provide services. We call such services infrastructure support. This Chapter will introduce commonly employed infrastructure support for secure communication systems.

### 1 Infrastructure Support

As we mentioned before, infrastructure support is a set of services to establish trust relationships. If the service is to authenticate a user, then a successful authentication protocol execution implies an established trust on the user's identity. Based on such a trust, the user under the claimed identity can be authorized, for example, to access a network or specific resource. If the service is to issue a certificate to a public key, then it is to establish a trust binding between a public key and its owner.

These services are provided based on a specific trust relationship. For example, an enterprise IT department may generate and distribute keys for the wireless devices used in the enterprise. On other hand, a service itself could be a business. For example, a certificate authority can be a commercial service provider to be trusted by different applications.

A service can be an online service, that is, the infrastructure provides service in real time in each execution of the security mechanism. Some can also be an offline service, that is, the interaction with the infrastructure is not needed at the time a security mechanism is executed.

The security requirements and trust models are quite different for each service. In this chapter, since most of the security mechanisms haven't been introduced formally, we may not be able to explain in details on how a security mechanism depends on a special service. But we feel that it is important to include general introduction in the early stage so that the readers can see the role of infrastructure support in a secure communication system. We will see the detailed interactions with each infrastructure support when the actual security mechanisms are introduced in the later chapters.

### 2 Authentication Server

An authentication server, as its name implies, is to provide authentication service. In order to explain the role of the authentication server, we will need to introduce entity authentication first.

#### 2.1 Entity Authentication

In a communication system, entity authentication is a process for one party to verify another party's identity with whom it communicates. A communication party is a node together with its controller, like a user. An entity authentication may be a user authentication or a device authentication. But in this section, we will use term entity or party without distinguish a user and a device. The

---

<sup>0</sup>Copyright ©2008 L.D. Chen and G. Gong. All rights reserved. May be freely reproduced for educational or personal use.

verification is to check whether the specific authentication data is valid, which can be generated only by the party with the claimed identity. In other words, the party to be authenticated must provide some verifiable evidence to prove it is the entity identified by the identity.

For example, for a given identity like user name, password is a widely used authentication evidence, which we probably have used every day. Password is generated by a user and stored in a place where it can be verified every time the authentication is conducted.

However, it is obvious that password can provide limited security for an entity authentication. Since human can only remember password with certain length, it is vulnerable to exhausting search attack. It is often the case that user generated passwords may not be completely random. As a result, the exhausting search can try the most likely password first to reduce the actual effort to succeed. A crucial limitation of the password based authentication is that when entering a password, it has to be protected from eavesdropping. Since entity authentication is often the first step in establishing protected communications, it may not be possible to protect the procedure of entering password.

Nevertheless, password based authentication illustrates some basic ideas about entity authentication. In this section, we will introduce a cryptographic authentication method. This method is often called challenge-response method. Basically, the method can be either public key based or symmetric key based. We will further introduce this method in Chapter 4. Here we use symmetric key based challenge-response method to explain the idea.

With symmetric key based method, assume both entities  $I$  and  $J$  share a key  $K_{IJ}$ . If party  $J$  will authenticate party  $I$ , party  $J$  will generate a random string  $Ch$ , called a challenge, and send to party  $I$ . Upon receiving the string  $Ch$ , party  $I$  generates a response  $Res$  by computing

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

where MAC is a message authentication code as we introduced before. In order to verify the response, party  $J$  will calculate

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

If  $Res = Res'$ , 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 1 illustrates an authentication by a challenge-response method.

We have noticed that in order to authenticate entity  $I$ , entity  $J$  must hold an authentication key and must have the capability to execute the cryptography operations. This leads to the need of an authentication server. We will explain in details in the next section.

## 2.2 Authentication Server

Entity authentication is often executed as an access authentication. That is, depending on the result of entity authentication, the entity can be authorized or rejected to access a network or certain information. For example, when a mobile phone is turned on, the first step is to register to the network for connections. The access permission depends on a subscriber authentication. If the authentication is successful, then the mobile phone gets connected, that is, it is authorized to access the cellular service.

In the above example of a mobile phone, the cellular network will authorize the access. In this section, the network entity which authorizes access is called a *network access server*. For example, an access points in wireless local area network plays the role of a network access server. Here the

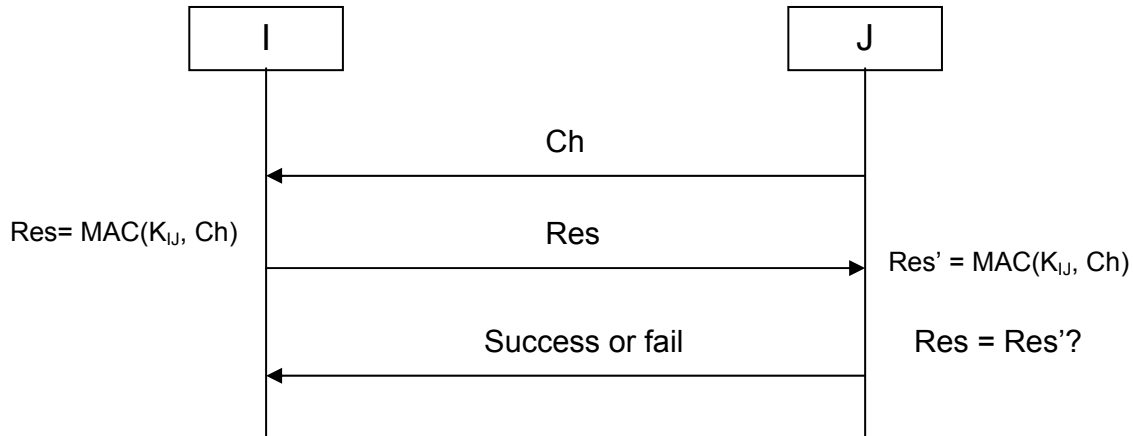


Figure 1: Entity Authentication by Challenge-Response

term server emphasizes the role in authorizing access for other entities. It does not indicate their processing capabilities nor physical security. As a matter of fact, it may not be physically safe to distribute the long-term keys used for authentication to these access servers. Furthermore, since the access may be requested and authorized through any of these access servers, centralized database and authentication will help for billing and other purpose. Therefore, this proposes a requirement to use a centralized database to execute authentication operations. As a result, an authentication server is often shared by many access servers. Not like each access server, an authentication server may be located remotely. Figure 2 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.

The authentication server introduced in this section is a general concept. We will see how it works with each special authentication protocol in Chapter 4.

### 3 Certificate Authority

The invention of public key cryptography in 1976 makes “secret communications” possible without distributing “secret keys”. A public key, as its name implies, is public and will not need to be distributed “secretly”. In the very famous example with two parties Alice and Bob, Alice can send secret information to Bob by simply encrypting the information with Bob’s public key. The Bob can decrypt it with Bob’s private key.

However, people quickly realized that the model with Alice and Bob can hardly be extended to a communication system with a large scale of users. The problem is how Alice can be insured that the key is Bob’s but not Eve’s public key. It may sound easy that Alice can get Bob’s public key through a conversation with Bob over the phone. But in a communication system, it is not realistic to assume the communication parties, even if they are human beings, know each other beforehand. Furthermore, such a key distribution method can hardly extend to a communication system.

In a public key cryptography scheme, it is feasible for any party to generate a pair of public and private keys. As a result, any one, for example, Eve, can generate such a key pair and to fool

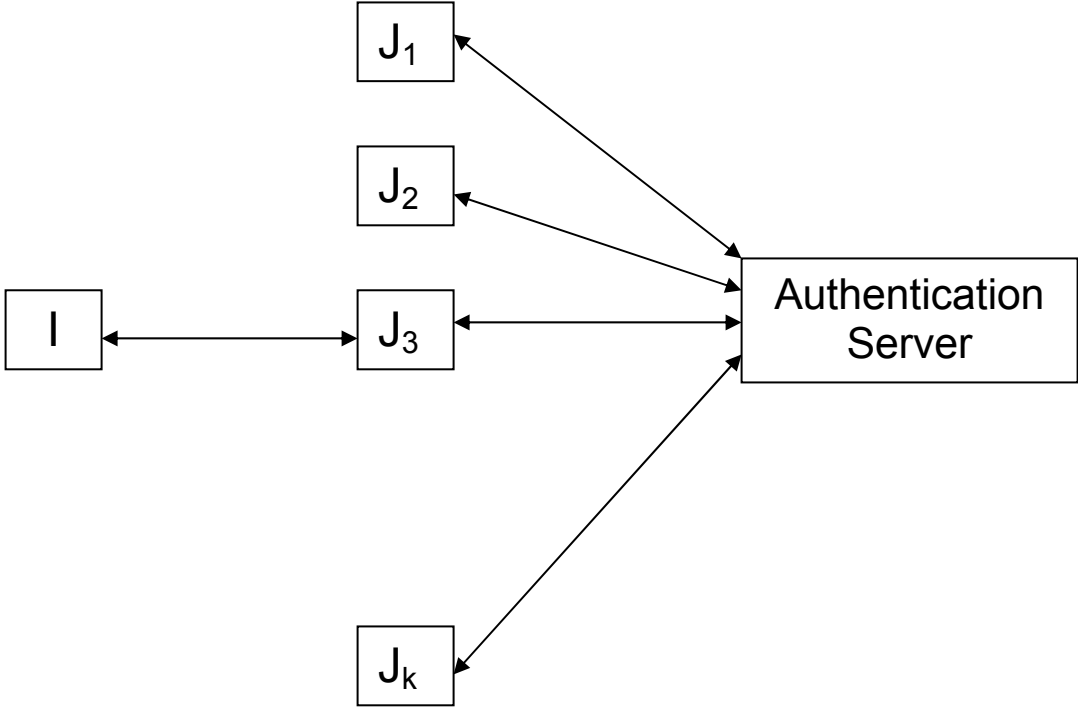


Figure 2: Authentication Server

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. As a result, instead of Bob, Eve can decrypt the information supposed to be accessed only by Bob. If this key pair is generated for digital signatures, then Eve can sign everything she wishes on the behalf of Bob.

Therefore, in order to provide the supposed security features with public key cryptography, it is crucial to bind a public key with its owner in a trusted way. For this reason, a trusted third party, called certificate authority (CA) is introduced to issue certificate.

### 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. Digital signature is used to insure such a binding. The CA will sign the public key together with its owner’s identity in order to generate a verifiable signature. 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. Indeed, it is crucial to insure that 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. Practically, each party can obtain the CA’s public key by downloading it from a trusted server. It can also install the CA’s public key in the communication devices, like Internet routers and personal computers.

Besides the public key and the identity, a certificate may include some other attributes, for example, the expiration date of the public key, the scope of the public key, i.e. whether it is used for encryption or signature, etc.

### 3.2 Certificate Chain and Revocation

Even though the main idea of introducing certificate authority is to convert peer-to-peer trust assumption to a multiple-to-one trust assumption, sometimes, it is impossible to depend on a single certificate authority to establish all the peer-to-peer trust relationship. For example, a system may consist of multiple administrations. Each administration has its own certificate authority,  $CA_1$  and  $CA_2$ , respectively. Two entities,  $I$  and  $J$ , belonging to two different administrations cannot establish trust unless two administrations have a commonly trusted third party. Therefore, in most of the practices, it needs a certificate hierarchy. For example, it may employ a root CA trusted by both administrations. We show this hierarchy in Figure 3.

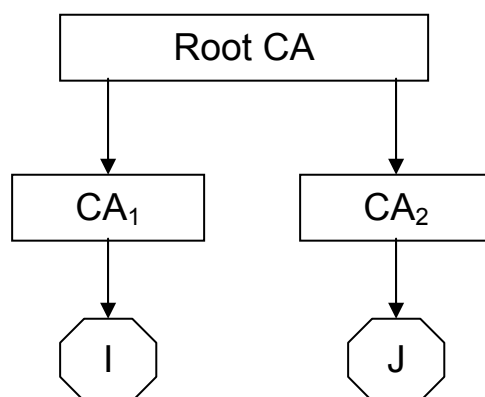


Figure 3: Certificate Hierarchy

In a certificate hierarchy, a higher level certificate authority will sign certificates of the lower level certificate authorities. In Figure 3, 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. Precisely,  $CA_1$  will sign entity  $I$ 's certificate, while  $CA_2$  will sign entity  $J$ 's certificate.

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.

Notice that the entity  $I$  does not have to have any trust relationship directly with  $CA_2$ , who signs  $J$ 's public key certificate. As long as entity  $I$  trusts the root CA's public key, then it can verify whether  $CA_2$ 's public key is trustable. Based on the verified trust on  $CA_2$ 's public key, it can verify whether  $J$ 's public key is trustable. This trust relationship can be described as a trust

chain. We also call it *a certificate chain*. It is illustrated in Figure 4. The arrow implies the action of using one public key to verify certificate of another public key as pointed.

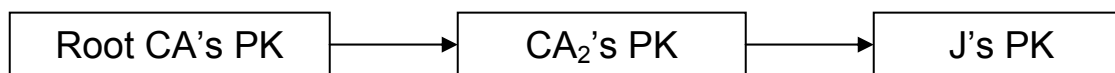


Figure 4: Certificate Chain

A certificate may have to be revoked before its expiration date. 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

Compared with the services provided by authentication server and certificate authority, the service a key generation server can provide must be restricted to the scenario that a security key distribution is available. Therefore, we assume that a key generation server is also a key distribution server. In this section, we will introduce key generation and distribution service as one of the infrastructure support.

### 4.1 Public/Private Key Pair Generation

When public key cryptography is introduced, the first distinctive advantage as we have mentioned is that it does not need to have a secure channel to distribute the key since a public key is public and no one except the owner needs to know the private key. However, practically, it is often that a public/private key pair is generated by a third party, for example, a certificate authority. In this case, the private key will need to be transported to the owner in a protected manner.

It may sound a little disappointing and inconvenient. But generating a public and private key pair often involve complicated mathematics operations. For example, it may need to search for primes or elliptic curves over a finite field with required properties. These operations may be very challenge to a low power device.

Another reason to depend on a third party to generate key pairs is to insure that the generated key should satisfy some designated security requirements for the devoted cryptographic operations. Some attacks discovered years ago are based on maliciously generated key pairs (see [1]).

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

Despite the historical debating on key escrow, the same term and concept is used for a service to provide back up for encrypted contents. A key escrow server 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.

However, a key generator may not be a key escrow server. For a key generator, it will not provide any back up service. Oppositely, it is required that the keys should be deleted from the server immediately after delivered to their owners for the obvious security reasons.

In Section 4.1, we discussed that a public/private key pair can be generated either by the owner or by a key generation server. However, some cryptography schemes must rely on a trusted third party to generate a key pair for each entity. The key generator must hold secret information on each key pair generated for an entity. For example, identity based encryption (IBE) is such a scheme (see [2]). In order to embed an identity to a public key, it relies on a trusted third party to generate the corresponding private key. Otherwise, it cannot embed the identity to a public key.

Notice that when an identity is embedded in a public key, they are bound together. Therefore, certificate authority is not needed for IBE schemes. Instead, the IBE schemes need a key generator. In either case, it depends on a trusted third party even though with different roles.

### 4.3 Symmetric Key Generation and Distribution

It is often that the strength of a symmetric key based cryptographic algorithm is measured by its key length. Since for an ideal cryptographic algorithm, the complexity of recovering the key should not be significantly lower than exhausting search. Therefore, the basic secure requirement of a symmetric key is its randomness. Obviously, it is not practical to generate a key by flipping a coin. A key should be generated by a qualified pseudorandom generator.

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. However, a system may only use symmetric key based cryptography mechanisms. In that case, without a key distribution server it is impossible to launch any security protection.

For symmetric keys, the key distribution may be accomplished in the manufacture by provisioning the keys to each device. It can also be distributed through service providers. However, 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.

## 5 Signing Server

Since any one with the certified public key can verify a digital signature without sharing the secret data, it brings a great convenience in authorizing and approving digital contents. For example, it can authorize a piece of software to be executed in a computer. It can also be used to sign copyrights on any digital products. In these applications, the signing procedure may be conducted by a server, called a *signing server*, and controlled by the authorities to issue the rights. This section will introduce a general description on signing servers. In Chapter 6, we will discuss the dependencies on signing servers in great details.

### 5.1 Signature for Authorized Software

Signing server is a kind of infrastructure support needed to build trusted platforms such that only authorized software can be executed on the platform. The software authorization is also used to make access control to certain data.

For software authorization, a signing server may be managed by a device manufacture or by an IT department of an enterprise. For the software running in subscriber devices, like cellular phones, operators may manage a signing server.

The interface with a signing server is often well controlled. For example, in a manufacture, the software must pass standard tests and approved by an authority before sending to the signing server.

For a given platform, software packages may be signed by different servers. They must be verified based on a trust hierarchy. We will get this into details in Chapter 6.

## 5.2 Signature for Copyrights

Today everything can be digitalized, from music to movies. Protection of digital copyrights has never been such challenging. Thank the invention of public key cryptography, digital signature provides a undisputable way to verify the digital copyrights.

The digital products can be signed by a signing server before being distributed to retailers. In Chapter 6, we will discuss how to enforce protection of digital copyrights in a trusted platform.

## References

- [1] R. Anderson and R. Needham, *Robustness Principle for Public Key Protocols*, *Advances in Cryptology – Crypto '95*, pp. 236–247, Lecture Notes in Computer Science, Volume 963, Springer-Verlag, 1995.
- [2] D. Boneh and M. Franklin. *Identity-Based Encryption from Weil Pairing*. *Advances in Cryptology- Crypto 2001* pp. 213–229. Lecture Notes in Computer Science, Volume 2139, Springer-Verlag. 2001.