UNIT-6 DIS

6.	Security in distributed systems, secure channels, authentication, integrity and confidentiality, access control, security management. Naming: Flat naming approaches, structured naming, name space and resolution, attribute- based naming, directory services, LDAP, decentralized implementations.	8
	TOTAL	**42**

Security in Distributed systems

Security in distributed systems can roughly be divided into two parts. One part concerns the communication between users or processes, possibly residing on different machines. The principal mechanism for ensuring secure communication is that of a secure channel. Secure channels, and more specifically, authentication, message integrity, and confidentiality, are discussed in a separate section.

The other part concerns authorization, which deals with ensuring that a process gets only those access rights to the resources in a distributed system it is entitled to. Authorization is covered in a separate section dealing with access control. In addition to traditional access control mechanisms, we also focus on access control when we have to deal with mobile code such as agents.

Secure Channels

9.2 Secure channels

The client-server model is a convenient way to organize a distributed system. In this model, servers may possibly be distributed and replicated, but also act as clients with respect to other servers. When considering security in distributed systems, it is once again useful to think in terms of clients and servers. In particular, making a distributed system secure essentially boils down to two predominant issues. The first issue is how to make the communication between clients and servers secure. Secure communication requires authentication of the communicating parties. In many cases it also requires ensuring message integrity and possibly confidentiality as well. As part of this problem, we also need to consider protecting the communication within a group of servers.

The second issue is that of authorization: once a server has accepted a request from a client, how can it find out whether that client is authorized to have that request carried out? Authorization is related to the problem of controlling access to resources.

The issue of protecting communication between clients and servers, can be thought of in terms of setting up a **secure channel** between communicating parties [Voydock and Kent, 1983]. A secure channel protects senders and receivers against interception, modification, and fabrication of messages. It does not also necessarily protect against interruption. Protecting messages against interception is done by ensuring confidentiality: the secure channel ensures that its messages cannot be eavesdropped by intruders. Protecting against modification and fabrication by intruders is done through protocols for mutual authentication and message integrity. A detailed and formal

CRYPTOGRAPHY AND AUTHENTICATION IS USED TO PROVIDE SECURITY TO DS.

Authentication

Authentication exists to establish and verify the identity of a user (or a service, process, or server). Once authentication is complete, a process can decide whether to allow access to the service or its resources and what type of access is permitted. This is called authorization.

The three factors of authentication are: *something you have* (such as a key or a card), *something you know* (such as a password or PIN), and *something you are* (biometrics).

Consequently, authentication and message integrity should go together. In many protocols, the combination works roughly as follows. Again, assume that Alice and Bob want to communicate, and that Alice takes the initiative in setting up a channel. Alice starts by sending a message to Bob, or otherwise to a trusted third party who will help set up the channel. Once the channel has been set up, Alice knows for sure that she is talking to Bob, and Bob knows for sure he is talking to Alice, they can exchange messages.

To subsequently ensure integrity of the data messages that are exchanged after authentication has taken place, it is common practice to use secret-key cryptography by means of session keys. A **session key** is a shared (secret) key that is used to encrypt messages for integrity and possibly also confidentiality. Such a key is generally used only for as long as the channel exists. When the channel is closed, its associated session key is destroyed.

Authentication based on a secret shared key .

Shared key authentication (SKA) is a verification method in which a computer or terminal uses the Wired Equivalent Privacy (WEP) protocol to access a wireless network. It pre-establishes that a requesting system has knowledge of a shared secret key required for authentication.

Shared key authentication (SKA) is not considered a secure method of granting network access because it uses conventional unsecured channels, like writing and verbal exchange, to share a security key for granting access.

Although the dissemination of the key is a large security issue, the authentication itself is secured using 64 or 128-bit encryption. It is difficult for an intruder to gain access without knowledge of the key.

SKA employs the following steps:

1. The requesting wireless device/client sends an identity assertion and authentication request to the access point (AP).
2. The access point challenges the client by sending a challenge text.
3. Using WEP and an encryption key, which is derived from the secret shared key (password), the client encrypts the challenge text and sends it back to the AP.
4. The AP decrypts the challenge text, and if it matches the one originally sent to the client, the authentication result is positive and the AP authenticates the client.
5. The client successfully connects to the network.

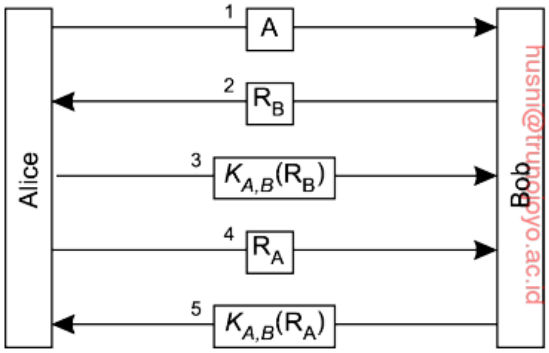

Figure 9.6: Authentication based on a shared secret key.

Authentication based on a key distribution center.

A key distribution center (KDC) in cryptography is a system that is responsible for providing keys to the users in a network that shares sensitive or private data. Each time a connection is established between two computers in a network, they both request the KDC to generate a unique password which can be used by the end system users for verification.

A key distribution center is a form of symmetric encryption that allows the access of two or more systems in a network by generating a unique ticket type key for establishing a secure connection over which data is shared and transferred. KDC is the main server which is consulted before communication takes place. Due to its central infrastructure, KDC is usually employed in smaller networks where the connection requests do not overwhelm the system. KDC is used instead of standard key encryption because the key is generated every time a connection is requested, which minimizes the chances of attack.

Authentication using a key distribution center

One of the problems with using a shared secret key for authentication is scalability. If a distributed system contains N hosts, and each host is required to share a secret key with each of the other $N-1$ hosts, the system as a whole needs to manage $N(N-1)/2$ keys, and each host has to manage $N-1$ keys. For large N, this will lead to problems. An alternative is to use a centralized approach by means of a **Key Distribution Center** (**KDC**). This KDC shares a secret key with each of the hosts, but no pair of hosts is required to have a shared secret key as well. In other words, using a KDC requires that we manage N keys instead of $N(N-1)/2$, which is clearly an improvement.

If Alice wants to set up a secure channel with Bob, she can do so with the help of a (trusted) KDC. The whole idea is that the KDC hands out a key to both Alice and Bob that they can use for communication, shown in Figure 9.9.

Alice first sends a message to the KDC, telling it that she wants to talk to Bob. The KDC returns a message containing a shared secret key $K_{A,B}$ that she can use. The message is encrypted with the secret key $K_{A,KDC}$ that Alice shares with the KDC. In addition, the KDC sends $K_{A,B}$ also to Bob, but now encrypted with the secret key $K_{B,KDC}$ it shares with Bob.

The main drawback of this approach is that Alice may want to start setting up a secure channel with Bob even before Bob had received the shared key from the KDC. In addition, the KDC is required to get Bob into the loop by passing him the key. These problems can be circumvented if the KDC just

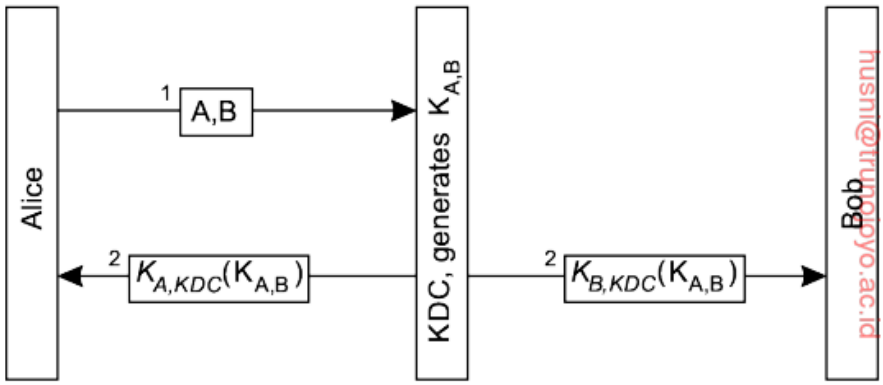

Figure 9.9: The principle of using a Key Distribution Center (KDC).

Authentication based on public-key cryptography

Public key cryptography (PKC) is an encryption technique that uses a paired public and private key (or asymmetric key) algorithm for secure data communication. A message sender uses a recipient's public key to encrypt a message. To decrypt the sender's message, only the recipient's private key may be used.

The two types of PKC algorithms are RSA, which is an acronym named after this algorithm's inventors: Rivest, Shamir and Adelman, and Digital Signature Algorithm (DSA). PKC encryption evolved to meet the growing secure communication demands of multiple sectors and industries, such as the military.

PKC is also known as public key encryption, asymmetric encryption, asymmetric cryptography, asymmetric cipher, asymmetric key encryption and Diffie-Hellman encryption.

Authentication using public-key cryptography

Let us now look at authentication with a public-key cryptosystem that does not require a KDC. Again, consider the situation that Alice wants to set up a secure channel to Bob, and that both are in the possession of each other's public key. A typical authentication protocol based on public-key cryptography is shown in Figure 9.13 which we explain next.

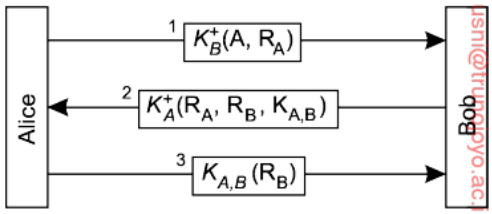

Figure 9.13: Mutual authentication in a public-key cryptosystem.

Alice starts with sending a challenge R_A to Bob encrypted with his public key K_B^+. It is Bob's job to decrypt the message and return the challenge to Alice. Because Bob is the only person that can decrypt the message (using the private key that is associated with the public key Alice used), Alice will know that she is talking to Bob. Note that it is important that Alice is guaranteed to be using Bob's public key, and not the public key of someone impersonating Bob. How such guarantees can be given is discussed later in this chapter.

When Bob receives Alice's request to set up a channel, he returns the decrypted challenge, along with his own challenge R_B to authenticate Alice. In addition, he generates a **session key** $K_{A,B}$ that can be used for further communication. Bob's response to Alice's challenge, his own challenge, and the session key are put into a message encrypted with the public key K_A^+ belonging to Alice, shown as message 2 in Figure 9.13. Only Alice will be capable of decrypting this message using her private key K_A^-.

Alice, finally, returns her response to Bob's challenge using the session key $K_{A,B}$ generated by Bob. In that way, she will have proven that she could decrypt message 2, and thus that she is actually Alice to whom Bob is talking.

Integrity and Confidentiality

- Message integrity means that messages are protected against surreptitious modification, confidentiality means that messages cannot be intercepted and read by eavesdroppers.
- Confidentiality can be achieved using encryption/decryption which can be performed using secret shared key or receiver's public key.
- Digital signatures:
    - It means associating a message with a unique signature that will help us to find whether someone modified our message.
    - Popular way of digital signing is using a public key cryptosystem such as RSA
- Session keys:
    - It is a unique key shared between parties during connection establishment of a secure channel, after the authentication phase has completed for confidentiality.

    - Reason for generating a unique key for each secure channel is to ensure protection against replay attacks(replaying an entire session again and again).
- Combination of long lasting keys(used in authentication) with cheaper and temporary session key is often used to achieve confidentiality and integrity

Access control

Access control means verifying access rights while authorization means grating access rights.

General issues in access control:

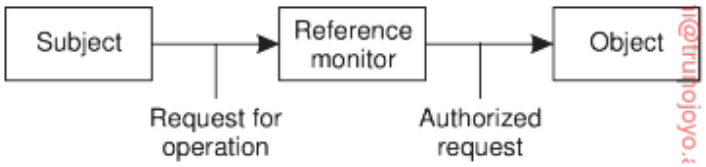

Figure 9.19: General model of controlling access to objects.

- Subject issues request to access an object.
- Controlling the access to an object is all about protecting the object against invocations by subjects that are not allowed to have specific (or even any) of the methods carried out. Also, protection may include object management issues, such as creating, renaming, or deleting objects. Protection is often enforced by a program called a **reference monitor**.
- It is extremely important that the reference monitor is itself tamperproof: an attacker must not be able to fool around with it.

Access control matrix:

- A common approach to modeling the access rights of subjects with respect to objects is to construct an **access control matrix**. Each subject is represented by a row in this matrix; each object is represented by a column. If the matrix is denoted M, then an entry M [s, o] lists precisely which operations subjects can request to be carried out on object o.

- Access control matrix is costly to use (same logic as graph, adjacency matrix and list). Therefore, another approach is to have each object maintain a list of the access rights of subjects that want to access the object. In essence, this means that the matrix is distributed column-wise across all objects, and that empty entries are left out. This type of implementation leads to what is called an **Access Control List (ACL).**
- Another approach is to distribute the matrix row-wise by giving each subject a list of **capabilities** it has for each object. In other words, a capability corresponds to an entry in the access control matrix. Not having a capability for a specific object means that the subject has no access rights for that object.
- An ACL or a capability list can still become quite large. One general way of reducing ACLs is to make use of **protection domains**. A protection domain is a set of (object, access rights), pairs. Each pair specifies for a given object exactly which operations are allowed to be carried out.

Firewalls:

External access to any part of a distributed system is controlled by a special kind of reference monitor known as a firewall.

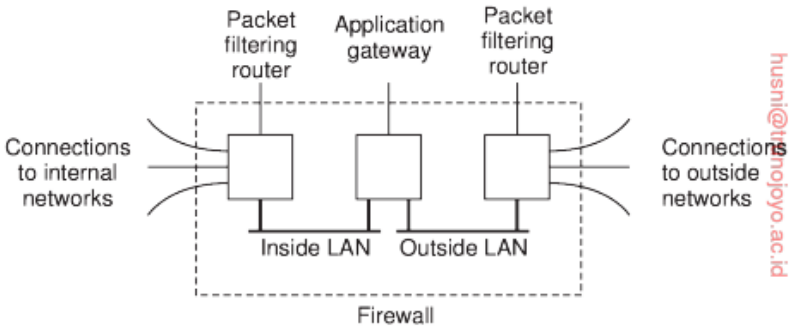

Figure 9.22: A common implementation of a firewall.

All outgoing, but especially all incoming packets are routed through a special computer and inspected before they are passed through. Unauthorized traffic is discarded and not allowed to continue.

Types of firewalls:

1. **Packet filtering gateway:** It operates as a router and makes decisions as to whether or not to pass a network packet based on the source and destination address as contained in the packet's header.
2. **Application level gateway:** In contrast to a packet-filtering gateway, this type of firewall actually inspects the content of an incoming or outgoing message.

Security management

Key Management:

1.1 Key establishment:

- A widely-applied scheme for establishing a shared key across an insecure channel is the **Diffie-Hellman key exchange.**
- **Diffie-Hellman key exchange:** It is a method of digital encryption that securely exchanges cryptographic keys between two parties over a public channel without their conversation being transmitted over the internet.
    - **Working:**

      To implement Diffie-Hellman, two end users, Alice and Bob, mutually agree on positive whole numbers p and q, such that p is a prime number and q is a generator of p. The generator q is a number that, when raised to positive whole-number powers less than p, never produces the same result for any two such whole numbers. The value of p may be large, but the value of q is usually small.

      Once Alice and Bob have agreed on p and q in private, they choose positive whole-number personal keys a and b. Both are less than the prime number modulus p. Neither user divulges their personal key to anyone; ideally, they memorize these numbers and don't write them down or store them anywhere. Next, Alice and Bob compute public keys a^* and b^* based on their personal keys according to the following formulas:

$a^* = qa \bmod p$

$b^* = qb \bmod p$

The two users can share their public keys a^* and b^* over a communications medium assumed to be insecure, such as the internet or a corporate wide area network. From these public keys, a number x can be generated by either user on the basis of their own personal keys. Alice computes x using the following formula:

$x = (b^*) \bmod p$

Bob computes x using the following formula:

$x = (a^*) \bmod p$

The value of x turns out to be the same according to either of the above two formulas. However, the personal keys a and b, which are critical in the calculation of x, haven't been transmitted over a public medium. Because it's a large and apparently random number, a potential hacker has almost no chance of correctly guessing x, even with the help of a powerful computer to conduct millions of trials. The two users can, therefore, in theory,communicate privately over a public medium with an encryption method of their choice using the decryption key x.

1.2 Key Distribution:

- In a symmetric cryptosystem, the initial shared secret key must be communicated along a secure channel that provides authentication as well as confidentiality.
- Public-key distribution takes place by means of public-key certificates. Such a certificate consists of a public key together with a string identifying the entity to which that key is associated.