University of Technology
الجامعة التكنولوجية
Computer Science Department
قسم علوم الحاسوب

Authentication and Access Control
التخويل والتحكم بالوصول

Lec.Prof.Dr.Ekhlas Khalaf
أ.د. اخلاص خلف

cs.uotechnology.edu.iq

Introduction

The goal of study of authentication, authorization, and access control is very important and interesting because these concepts form the backbone of secure information systems, they ensure that only the right users have access to the right resources at the right time, helping to protect data and resources from unauthorized access, attackers, and security threats.

Definition: -

Authentication: - is the process of verifying the identity of a user, device, or system before granting access to resources, systems, or services. It ensures that the entity requesting access is indeed who or what it claims to be. Authentication typically involves presenting one or more credentials, such as a password, biometric data, or a security token, which are then compared against stored data to confirm the entity's identity.

Authentication provides a method to identify users, which includes the login and password dialog, challenge and response, messaging support, and encryption, depending on the selected security protocol. Authentication is the way a user is identified prior to being allowed access to the network and network services. Authentication is just one step in ensuring only the right users perform the right actions

Authentication methods: -

- **Security token**

security token is a peripheral device used to gain access to an electronically restricted resource. The token is used in addition to or in place of a password. It acts like an electronic key to access something. Examples include a wireless keycard opening a locked door, or in the case of a customer trying to access their bank account online, the use of a bank-provided token can prove that the customer is who they claim to be.

- **Passphrase**

A passphrase is a sequence of words or other text used to control access to acomputer system, program, or data. A passphrase is similar to a password in usage but is generally longer for added security. Passphrases are often used to control both access to, and operation of, cryptographic programs and systems, especially those that derive an encryption key from a passphrase.

- **Keystroke logging**

Keystroke logging, often referred to as keylogging or keyboard capturing, is the action of recording (logging) the keys struck on a keyboard, typically covertly, so that person using the keyboard is unaware that their actions are being monitored Data can then be retrieved by the person operating the logging program.A keylogger can be either software or hardware.

- **Challenge–response authentication**

Challenge–response authentication is a family of protocols in which one party presents a question ("challenge") and another party must provide a valid answer ("response") to be authenticated. The simplest example of a challenge–response protocol is password authentication, where the challenge is asking for the password and the valid response is the correct password.

- **Software token**

A software token is a piece of a two-factor authentication security device that may be used to authorize the use of computer services. Software tokens are stored on a general-purpose electronic device such as a desktop computer, laptop, PDA,or mobile phone and can be duplicated.

- **Shoulder surfing**

shoulder surfing is a type of social engineering technique used to obtain information such as personal identification numbers (PINs), passwords and other confidential data, either from keystrokes on a device or sensitive informationbeing spoken and heard, also known as eavesdropping (is the act of secretly or stealthily listening to the private conversation or communications of others without their consent)

- **Partial password**

A partial password is a mode of password authentication intended to make keystroke logging and shoulder surfing less effective. By asking the user to enter only a few specific characters from their password, rather than the whole password, partial passwords help to protect the user from password theft.

Authentication Factors

The ways in which someone may be authenticated fall into three categories, based on what are known as the factors of authentication: something the user knows, something the user has, and something the user is.

Each authentication factor covers a range of elements used to authenticate orverify a person's identity prior to being granted access, approving a transaction request, signing a document or other work product, granting authority to others, and establishing a chain of authority.

The Three factors (classes) and some of elements of each factor are:

1. The knowledge factors: Something the user knows (e.g., a password, partial password, pass phrase, or personal identification number (PIN), challenge response (the user must answer a question, or pattern), Security question.
2. The ownership factors: Something the user has (e.g., wrist band, ID card, security token, cell phone with built-in hardware token, software token, or cell phone holding a software token)
3. The inherence factors: Something the user is or does (e.g., fingerprint, retinal pattern, DNA sequence, signature, face, voice, unique bio-electric signals, or other biometric identifier)

Authentication types

The most frequent types of authentications available in use for authenticating online users differ in the level of security provided by combining factors from the one or more of the three categories of factors for authentication:

- ✓ **Single Factor Authentication**

  Also known as primary authentication, this is the simplest and most common form of authentication. Single Factor Authentication requires, of course, only one authentication method such as a password, security pin, etc. to grant access to a system or service.

  While these methods score high on usability and familiarity, they are typically associated with poor security and can be easily guessed or stolen via data breaches, phishing or using keyloggers.

- **2nd Factor Authentication**

  Adding a layer of complexity, 2FA requires a second factor to verify a user's identity. Common examples include tokens generated by a registered device, One Time Passwords, or PIN numbers. The mere presence of two authentication methods improves your security posture significantly.

- **Multi-Factor Authentication**

  Multi-Factor Authentication (MFA) is the most sophisticated authentication method that leverages 2 or more independent factors to grant user access to a system. In typical scenarios, MFA methods leverage at least 2 or 3 of the following categories:

  - Something you know - a password or a pin
  - Something you have - mobile phone or a security token
  - Something you are - fingerprint or FaceID
  - Something you do - typing speed, locational information etc.

User Authentication

A user authentication policy is a process in which you verify that someone who is attempting to access services and applications is who they claim to be. This can be accomplished through a variety of authentication methods, such as entering a password into your laptop or phone or a PIN number into the ATM.

What are the different authentication protocols?

Network authentication protocols are used to help securely transfer identity credentials for authentication between the subject (user or device) and the authentication server. There are several different authentication protocols for network access control, including:

1. Kerberos
2. Extensible Authentication Protocol (EAP)
3. IEEE 802.1X
4. Remote Authentication Dial-In User Service (RADIUS)
5. Terminal Access Controller Access-Control System (TACACS)

Message Authentication

Message authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent by (i.e., contain no modification, insertion, deletion, or replay) and that the purported identity of the sender is valid. One of the most fascinating and complex areas of cryptography is that of message authentication and the related area of digital signatures. There are two main components of message authentication: -

1. **Message integrity:** - Ensure that the message has not been changed or altered during transmission, any alteration in the message will be detected by the receiver.
2. **Message authenticity**: - confirms that the message indeed originated from the claimed sender and not from an impersonator.

Message Authentication Functions

Any message authentication or digital signature mechanism has two levels of functionality. At the lower level, there must be some sort of function that produces an authenticator**:** a value to be used to authenticate a message. This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

The types of functions that may be used to produce an authenticator. These may be grouped into three classes.

1. Hash function: A function that maps a message of any length into a fixed length hash value, which serves as the authenticator.

2. Message encryption: The ciphertext of the entire message serves as its authenticator

3. Message Authentication Code (MAC): A function of the message and a secretkey that produces a fixed-length value that serves as the authenticator.

Message Encryption

Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.

SYMMETRIC ENCRYPTION Consider the straightforward use of symmetric encryption Figure **(12.1a).** A message transmitted from source A to destination B is encrypted using a secret key shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.

In addition, B is assured that the message was generated by A. Why? The message must have come from A, because A is the only other party that possesses K and therefore the only other party with the information necessary to construct ciphertext that can be decrypted with K. Furthermore, if M is recovered, B knows that none of the bits of M have been altered, because an opponent that does not know K would not know how to alter bits in the ciphertext to produce the desired changes in the plaintext.

So we may say that symmetric encryption provides authentication as well as confidentiality. However, this flat statement needs to be qualified. Consider exactly what is happening at B. Given a decryption function D and a secret key K, the destination will accept any input X and produce output Y = D(K, X) . If X is the ciphertext of a legitimate message M produced by the corresponding encryption function, then Y is some plaintext message M. Otherwise, Y will likely be a meaningless sequence of bits. There may need to be some automated means of determining at B whether Y is legitimate plaintext and therefore must have come from A.

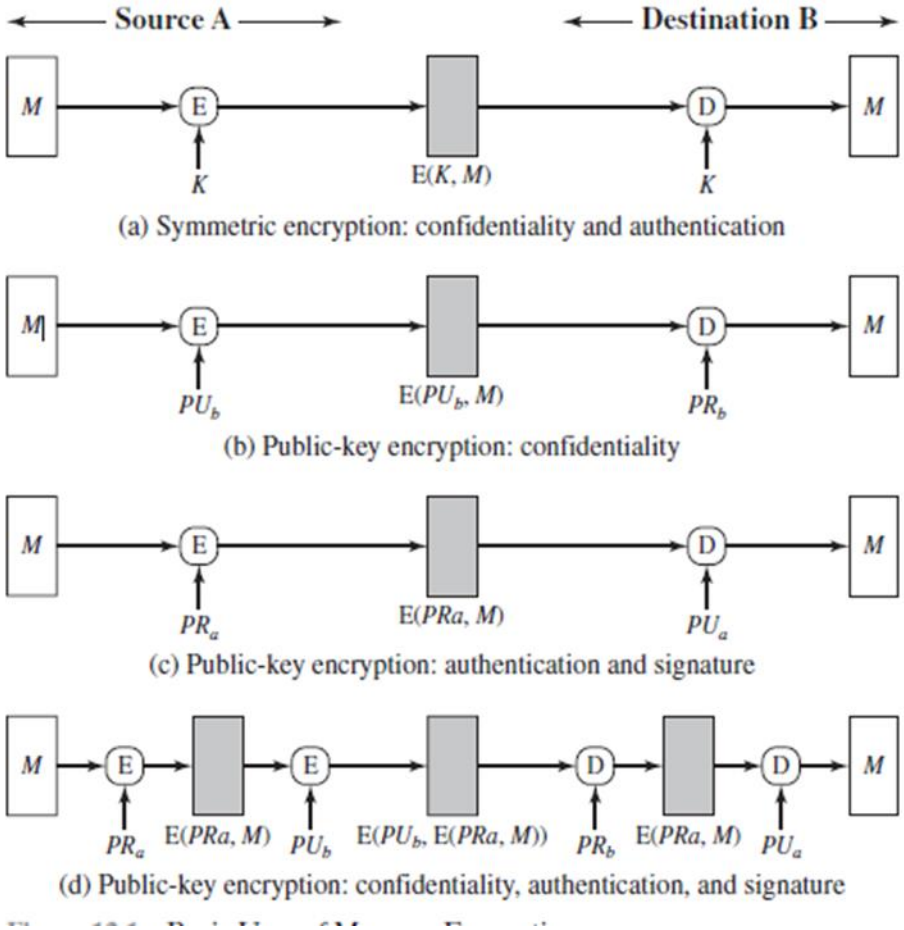


Figure (12.1) Basic Uses of Message Encryption

The implications of the line of reasoning in the preceding paragraph are profound from the point of view of authentication. Suppose the message M can be any arbitrary bit pattern. In that case, there is no way to determine automatically, at the destination, whether an incoming message is the ciphertext of a legitimate message.

This conclusion is incontrovertible: If M can be any bit pattern, then regardless of the value of X, the value Y = D(K, X) is bit pattern and therefore must be acceptedas authentic plaintext.

For a number of applications and encryption schemes, the desired conditions prevail as a matter of course. For example, suppose that we are transmitting English language messages using a

Caesar cipher with a shift of one (K=1). **A sends the following legitimate ciphertext:**

nbsftfbupbutboeepftfbupbutboemjuumfmbnctfbujwz

B decrypts to produce the following plaintext:

mareseatoatsanddoeseatoatsandlittlelambseativy

A simple frequency analysis confirms that this message has the profile of ordinary English. On the other hand, if an opponent generates the following random sequence of letters:

zuvrsoevgqxlzwigamdvnmhpmccxiuureosfbcebtqxsxq

This decrypts to

ytuqrndufpwkyvhfzlcumlgolbbwhttqdnreabdaspwrwp

which does not fit the profile of ordinary English.

PUBLIC-KEY ENCRYPTION: The straight forward use of public-key encryption **(Figure 12.1b)** provides confidentiality but not authentication. The source (A) uses the public key PU_b of the destination (B) to encrypt *M*. Because only B has the corresponding private key PR_b, only B can decrypt the message. This scheme provides no authentication, because any opponent could also use B's public key to encrypt a message and claim to be A.

To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt **(Figure 12.1c).**This provides authentication using the same type of reasoning as in the symmetric encryption case: The message must have come from A because A is the only party that possesses *PRa* and therefore the only party with the information necessary to construct ciphertext that can be decrypted with PUa . Again, the same reasoning as before applies: There must be some internal structure to the plaintext so that the receiver can distinguish between well-formed plaintext and random bits.

Assuming there is such structure, then the scheme of Figure **12.1c** does provide authentication. It also provides what is known as digital signature.1 Only A could have constructed the ciphertext because only A possesses *PRa* . Not even B, the recipient, could have constructed the ciphertext. Therefore, if B is in possession of the ciphertext, B has the means to prove that the message must have come from A. In effect, A has "signed" the message by using its private key to encrypt. Note that this scheme does not provide confidentiality. Any one in possession of A's public key can decrypt the ciphertext.

To provide both confidentiality and authentication, A can encrypt *M* first using its private key, which provides the digital signature, and then using B's public key, which provides confidentiality **(Figure 12.1d).** The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

Digital Signature

A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. Typically, the signature is formed by taking the hash of the message and encrypting the message with the creator's private key. The signature guarantees the work source and integrity of the message.

The most important development from the on public-key cryptography is the digital signature. The digital signature provides a set of security capabilities that would be difficult to implement in any other way.

Figure 13.1 is a generic model of the process of making and using digital signatures. Bob can sign a message using a digital signature generation algorithm. The inputs to the algorithm are the message and Bob's private key. Any other user, say Alice, can verify the signature using a verification algorithm, whose inputs are the message, the signature, and Bob's public key.

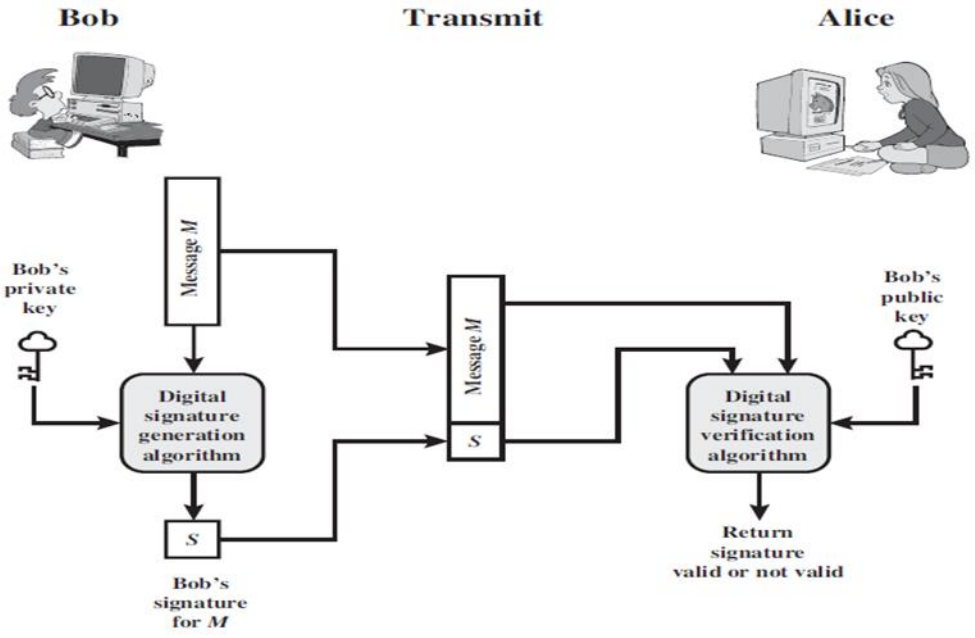


Figure (13.1) Generic Model of Digital Signature Process

In simplified terms, the essence of the digital signature mechanism is shown in **Figure 13.2**.

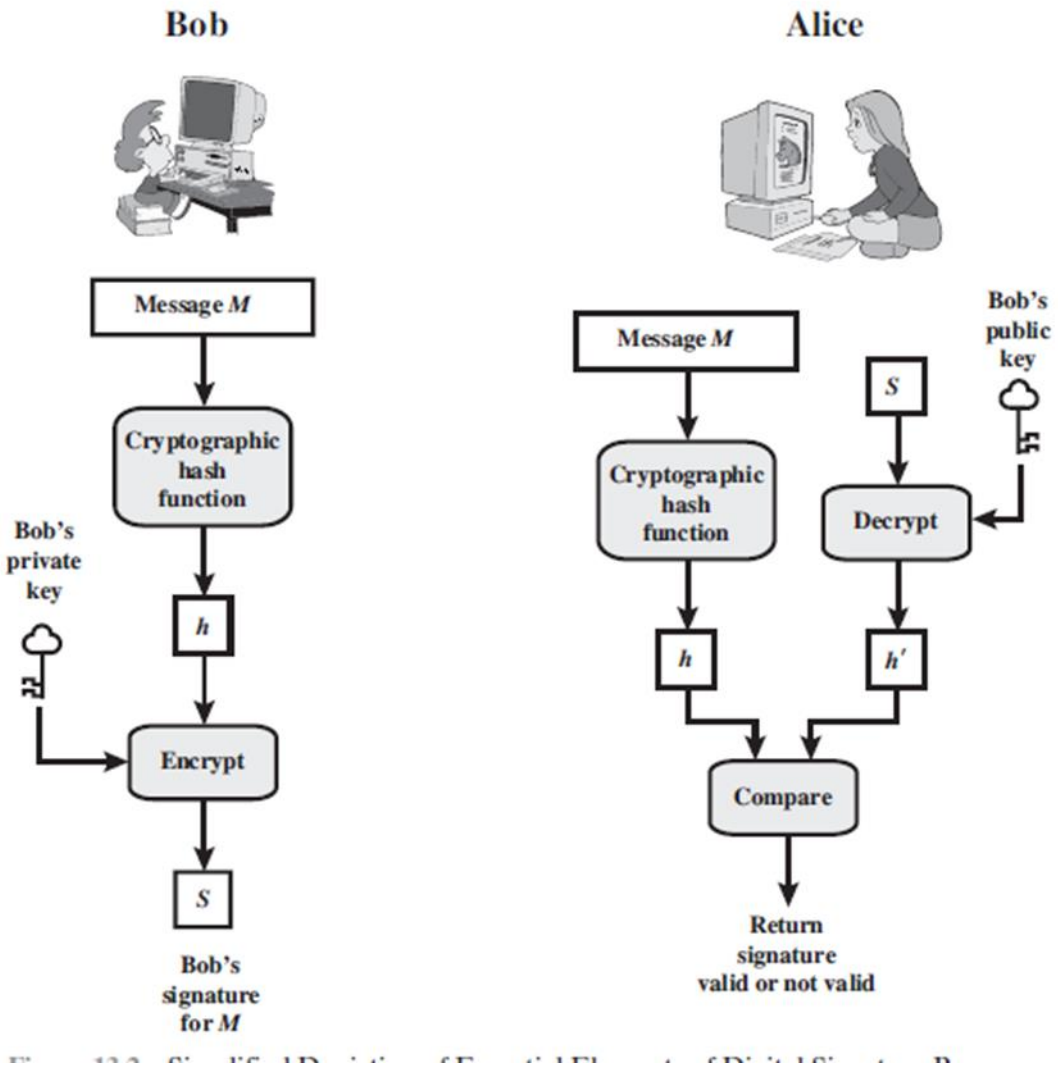


Figure 13.2 Simplified Depiction of Essential Elements of Digital Signature

Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other.Several forms of dispute between the two are possible.

For example, suppose that John sends an authenticated message to Mary, using one of the schemes of Figure 12.1. Consider the following disputes that could arise.

1. Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.

2. John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

Both scenarios are of legitimate concern. Here is an example of the first scenario: An electronic funds transfer takes place, and the receiver increases the amount of funds transferred and claims that the larger amount had arrived from the sender. An example of the second scenario is that an electronic mail message contains instructions to a stockbroker for a transaction that subsequently turns out badly. The sender pretends that the message was never sent.

In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature. **The digital signature must have the following properties:**

- ✓ It must verify the author and the date and time of the signature.
- ✓ It must authenticate the contents at the time of the signature.
- ✓ It must be verifiable by third parties, to resolve disputes.

Thus, the digital signature function includes the authentication function.

Mutual authentication

An important application area is that of mutual authentication protocols. Such protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys. There, the focus was key distribution. We return to this topic here to consider the wider implications of authentication.

Central to the problem of authenticated key exchange are two issues: confidentiality and timeliness. To prevent masquerade and to prevent compromise of session keys, essential identification and session-key information must be communicated in encrypted form. This requires the prior existence of secret or public keys that can be used for this purpose. The second issue, timeliness, is important because of the threat of message replays. Such replays, at worst, could allow an opponent to compromise a session key or successfully impersonate another party. At minimum, a successful replay can disrupt operations by presenting parties with messages that appear genuine but are not.

Examples of Replay Attacks:

- ✓ Simple replay: The opponent simply copies a message and replays it later.
- ✓ Repetition that can be logged: An opponent can replay a timestamped message within the valid time window.
- ✓ Repetition that cannot be detected: This situation could arise because the original message could have been suppressed and thus did not arrive at its destination; only the replay message arrives.
- ✓ Backward replay without modification: This is a replay back to the message sender. This attack is possible if symmetric encryption is used, and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

One approach to coping with replay attacks is to attach a sequence number to each message used in an authentication exchange. A new message is accepted only if its sequence number is in the proper order. The difficulty with this approach is that it requires each party to keep track of the last sequence number for each claimant it has dealt with. **Because of this overhead, sequence numbers are generally not used for authentication and key exchange. Instead, one of the following two general approaches is used:**

1. **Timestamps:** Party A accepts a message as fresh only if the message contains a timestamp that, in A's judgment, is close enough to A's knowledge of current time. This approach requires that clocks among the various participants be synchronized.

2. **Challenge/response:** Party A, expecting a fresh message from B, first sends B a nonce (challenge) and requires that the subsequent message (response) received from B contain the correct nonce value.

Remote User Authentication using symmetric encryption

Remote user authentication is a mechanism in which the remote server verifies the legitimacy of a user over an insecure communication channel. Password based authentication schemes have been widely deployed to verify the legitimacy of remote users as password authentication is one of the simplest and the most convenient authentication mechanism over insecure networks.

For example, user Alice Toklas could have the user identifier ABTOKLAS, This information needs to be stored on any server or computer system that Alice wishes to use and could be known to system administrators and other users.A typical item of authentication information associated with this user ID is a password, which is kept secret (known only to Alice and to the system).

obtain or guess Alice's password, then the combination of Alice's user ID and password enables administrators to set up Alice's access permissions and it her activity. Because Alice's ID is not secret, system users can send her e-mail, but because her password is secret, no one can pretend to be Alice.

The process of verifying an identity claimed by or for a system entity. An authentication process consists of two steps:

- **Identification step:** Presenting an identifier to the security system. (Identifiers should be assigned carefully, because authenticated identities are the basis for other security services, such as access control service.)
- **Verification step:** Presenting or generating authentication information that corroborates the binding between the entity and the identifier.

There are four general means of authenticating a user's identity, which can be used alone or in combination:

1. Something the individual knows: Examples include a password, a personal identification number (PIN), or answers to a prearranged set of questions.
2. Something the individual possesses: Examples include cryptographic keys, electronic keycards, smart cards, and physical keys. This type of authenticator is referred to as a token.
3. Something the individual is (static biometrics): Examples include recognitionby fingerprint, retina, and face.
4. Something the individual does (dynamic biometrics): Examples include recognition by voice pattern, handwriting characteristics, and typing rhythm.

The most important mechanisms in authentication are x.509 ad Kerberos, they are different but both of them used for authentication as explained in the following sections:-

❖ X.509 Standard

X.509 is a standard format for public key certificates, which are used in various security protocols to verify the identity of entities (such as websites, users or devices) and to establish secure communications over networks like the internet.X.509 defines the format for public-key certificates. This format is widely used in a variety of applications.

X.509 is part of the X.500 series of recommendations that define a directory service. The directory is, in effect, a server or distributed set of servers that maintains a database of information about users. The information includes a mapping from user name to network address, as well as other attributes and information about the users.

X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates of the type. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates. X.509 is an important standard because the certificate structure

and authentication protocols defined in X.509 are used in a variety of contexts.

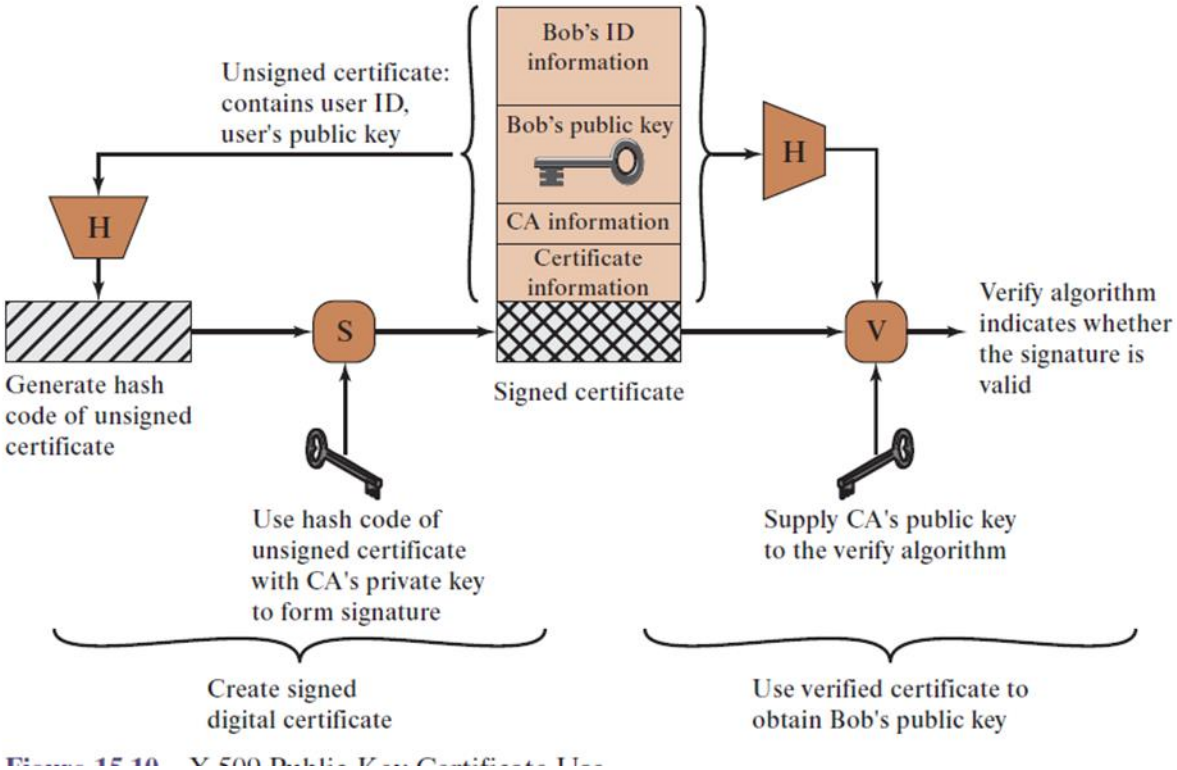
- ✓ X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends RSA. The digital signature scheme is assumed to require the use of a hash function.
- ✓ One scheme has become universally accepted for formatting public-key certificates: the X.509 standard. X.509 certificates are used in most network security applications, including IP security, transport layer security (TLS), and S/MIME,
- ✓ Typically, public-key infrastructure (PKI) implementations make use of X.509 certificates.

❖ X.509 Certificates

X.509 certificates are a standard format for public key certificates which reused in various security protocols to establish a secure and authenticated connection between parties over networks

The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user. The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

Figure shows the general format of a certificate, which includes the following elements.



Figure/ general format of a x.509 certificate

The elements of X.509 certificates include the following elements; -

- **Version:** Differentiates among successive versions of the certificate format;
  1. the default is version 1.
  2. If the issuer unique identifier or subject unique identifier are present, the value must be version 2.
  3. If one or more extensions are present, the version must be version 3.
- **Serial number:** An integer value unique within the issuing CA that is unambiguously associated with this certificate.
- **signature algorithm identifier:** The algorithm used to sign the certificate together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.

- **Issuer name:** X.500 is the name of the CA that created and signed this certificate.
- **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.
- **Subject name:** The name of the user to whom this certificate refers.That is, this certificate certifies the public key of the subject who holds the corresponding private key.
- **Subject's public-key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.
- **Issuer unique identifier:** An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.

❖ Kerberos

Kerberos is an authentication service designed for use in a distributed environment. Kerberos provides a trusted third-party authentication service that enables clients and servers to establish authenticated communication.

Kerberos is an authentication service developed as part of Project Athena at MIT. The problem that Kerberos addresses is this: Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network. We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services.

In particular, the following three threats exist:

1. A user may gain access to a particular workstation and pretend to be another user operating from that workstation.

2. A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.

3. A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access. Rather than building in elaborate authentication protocols at each server, Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Unlike most other authentication schemes described in this book, Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption.

❖ Kerberos Motivation

If a set of users is provided with dedicated personal computers that have nonetwork connections, then a user's resources and files can be protected by physically securing each personal computer. When these users instead are served by a centralized timesharing system, the time-sharing operating system must provide the security. The operating system can enforce access-control policiesbased on user identity and use the logon procedure to identify users.

Today, neither of these scenarios is typical. More common is a distributed architecture consisting of dedicated user workstations (clients) and distributed or centralized servers. In this environment, **three approaches to security can be envisioned.**

1. Rely on each individual client workstation to assure the identity of its user orusers and rely on each server to enforce a security policy based on user identification (ID).
2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.
3. Require the user to prove his or her identity for each service invoked. Alsorequire that servers prove their identity to clients.

Transport Layer Security (TLS) and Secure Sockets Layer (SSL)

TLS and SSL are cryptographic protocols designed to provide communications security over a computer network. Several versions of the protocols find widespread use in applications such as web browsing, email, instant messaging,

and voice over IP (VoIP). Websites can use TLS to secure all communications between their servers and web browsers.

- ✓ Secure Socket Layer (SSL) provides security services between TCP and applications that use TCP.The Internet standard version is called Transport Layer Service (TLS).
- ✓ SSL/TLS provides confidentiality using symmetric encryption and message integrity using a message authentication code(MAC).
- ✓ SSL/TLS includes protocol mechanisms to enable two TCP users to determine the security mechanisms and services they will use.
- ✓ HTTPS (HTTP over SSL) refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server.
- ✓ Secure Shell (SSH) provides secure remote logon and other secure client/server facilities.

Web Traffic Security Approaches

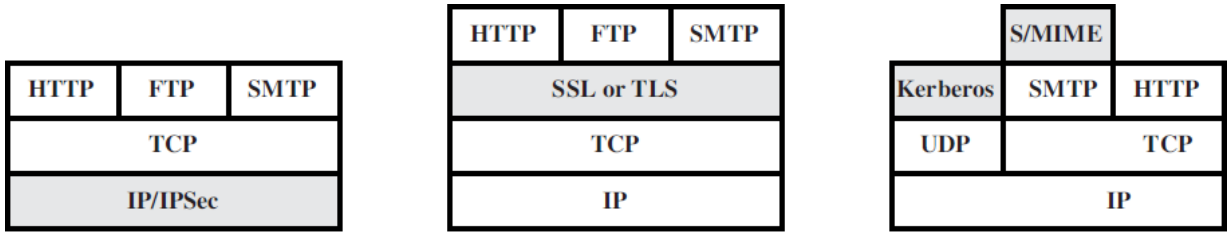
Several approaches to providing Web security are possible. The various approachesthat have been considered are similar in the services they provide and, to some extent, in the mechanisms that they use, but they differ with respect to their scopeof applicability and their relative location within the TCP/IP protocol stack.

One way to provide Web security is to use IP security (IPsec). The advantage of using IPsec is that it is transparent to end users and applications and provides a general-purpose solution. Furthermore, IPsec includes a filtering capability so that only selected traffic need incur the overhead of IPsec processing.

The foremost example of this approach is the Secure Sockets Layer (SSL) and the follow-on Internet standard known as Transport Layer Security (TLS). At this level, there are two implementation choices. For full generality, SSL (or TLS) could be provided as part of the underlying protocol suite and therefore be transparent to applications. Alternatively, SSL can be embedded in specific packages. For example,

Netscape and Microsoft Explorer browsers come equipped with SSL, and most Web servers have implemented the protocol.

Application-specific security services are embedded within the particular application. Figure below shows examples of this architecture. The advantage of this approach is that the service can be tailored to the specific needs of a given application.



(a) Network level (b) Transport level (c) Application level

SSL Architecture

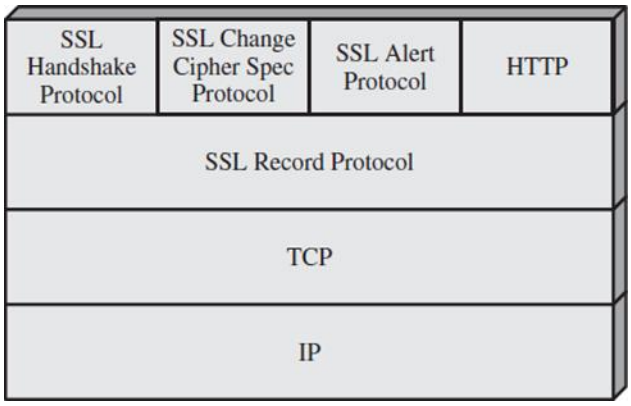
SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol but rather two layers of protocols, as illustrated inFigure below.

The SSL Record Protocol provides basic security services to various higher layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides thetransfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: the Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol. These SSL-specific protocols are used in the management of SSL exchanges and are examined later in this section.

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows.

Connection: A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.

Session: An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.



Between any pair of parties (applications such as HTTP on client and server), there may be multiple secure connections. In theory, there may also be multiple simultaneous sessions between parties, but this feature is not used in practice.

There are a number of states associated with each session. Once a session is established, there is a current operating state for both read and write (i.e., receive and send). In addition, during the Handshake Protocol, pending read and write states are created. Upon successful conclusion of the Handshake Protocol, the pending states become the current states.

A session state is defined by the following parameters:-

- Session identifier: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
- Peer certificate: An X509.v3 certificate of the peer. This element of the state may be null.

- Compression method: The algorithm used to compress data prior to encryption.
- Cipher spec: Specifies the bulk data encryption algorithm (such as null,AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation.It also defines cryptographic attributes such as the hash_size.
- Master secret:48-byte secret shared between the client and server.
- Is resumable: A flag indicating whether the session can be used to initiate new connections.

A connection state is defined by the following parameters.

- **Server and client random**: Byte sequences that are chosen by the server and client for each connection.
- **Server write MAC secret**: The secret key used in MAC operations on data sent by the server.
- **Client write MAC secret**: The secret key used in MAC operations on datasent by the client.
- **Server write key**: The secret encryption key for data encrypted by the server and decrypted by the client.
- **Client write key**: The symmetric encryption key for data encrypted by the client and decrypted by the server.
- **Initialization vectors**: When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter, the final ciphertext block from each record is preserved for use as the IV with the following record.
- **Sequence numbers**: Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed $2^{64} - 1$.

SSL Record Protocol

The SSL Record Protocol provides two services for SSL connections:

- ✓ **Confidentiality**: The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
- ✓ **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

Figure **17.3** indicates the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled beforebeing delivered to higher-level users.

The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 2^{14} bytes (16384 bytes) or less. Next, compression is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes.1In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null.

The next step in processing is to compute **a message authentication code** over the compressed data. For this purpose, a shared secret key is used. The calculation is defined as

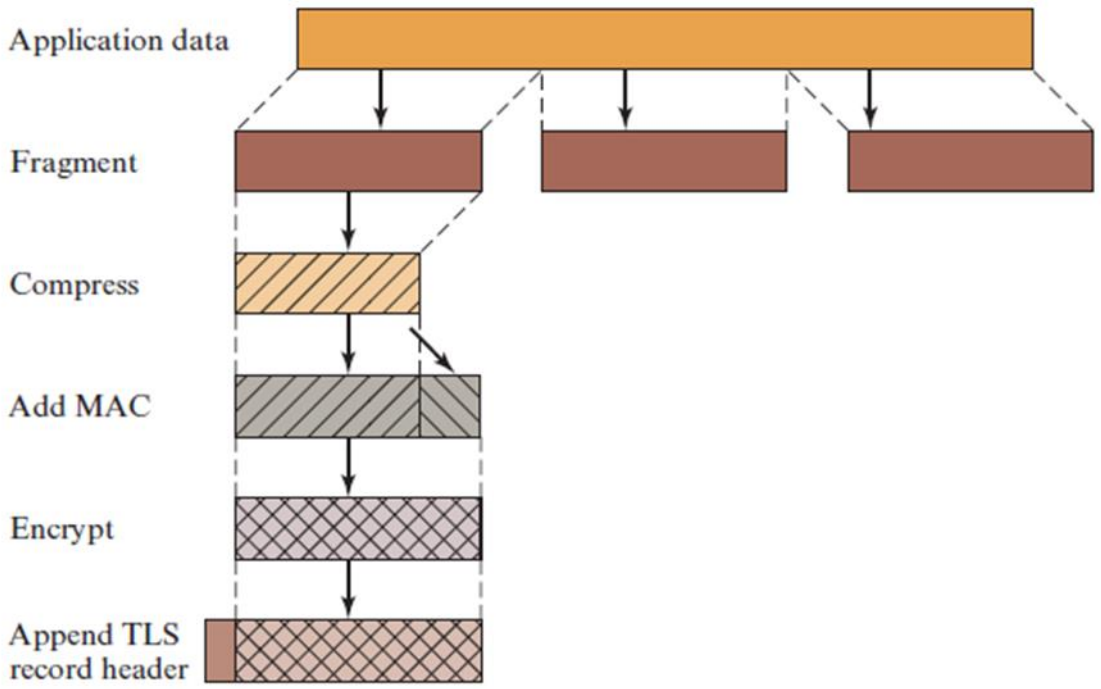


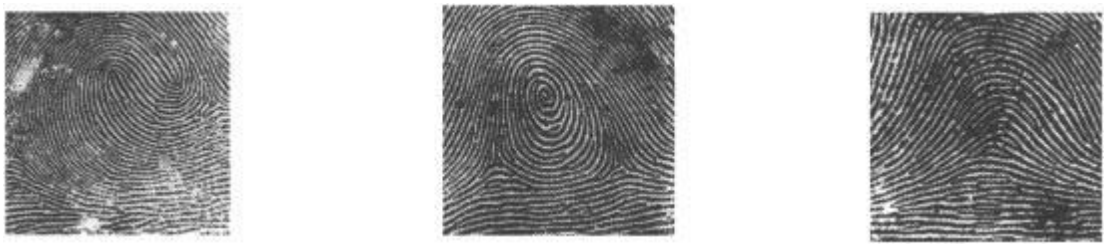
Figure 17.3 TLS Record Protocol Operation

Biometrics Methods

1. Fingerprints

Fingerprints were used in ancient China as a form of signature, and they have served a similar purpose at other times in history. But the use of fingerprints as a scientific form of identification is a much more recent phenomenon.

A fingerprint biometric works by first capturing an image of the fingerprint. The image is then enhanced using various image-processing techniques, and various points are identified and extracted from the enhanced image.



Figure/fingerprints examples

The following steps explain the process of FINGERPRINTS

1. fingerprint scanner (optical, capacitive, or ultrasonic) is used to capture an image of the fingerprint.
2. Preprocessing: removing noise, enhancing contrast, and clarifying the ridges and valleys. This step ensures that the fingerprint is clear enough for accurate feature extraction.
3. Feature Extraction: The system identifies key features of the fingerprint, known as minutiae points (e.g., ridge endings, bifurcations).The system converts these minutiae points into a digital template that represents the unique aspects of the fingerprint.
4. Matching: When the user attempts to authenticate, their fingerprint is scanned again, and a new template is generated. The system compares the new template with the stored one using pattern-matching algorithms to find similarities between the minutiae points.

2. Iris Scan

one of the best for authentication is the iris scan. The development of the iris (the colored part of the eye) is chaotic, which implies that minor variations lead to large differences. There is little or no genetic influence on the iris pattern, resulting in a unique pattern for each person, this pattern is highly stable over a person's life.

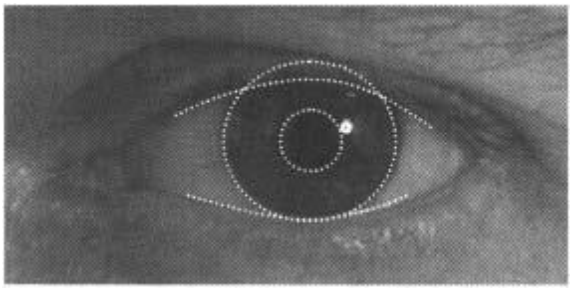


Figure () Iris scan

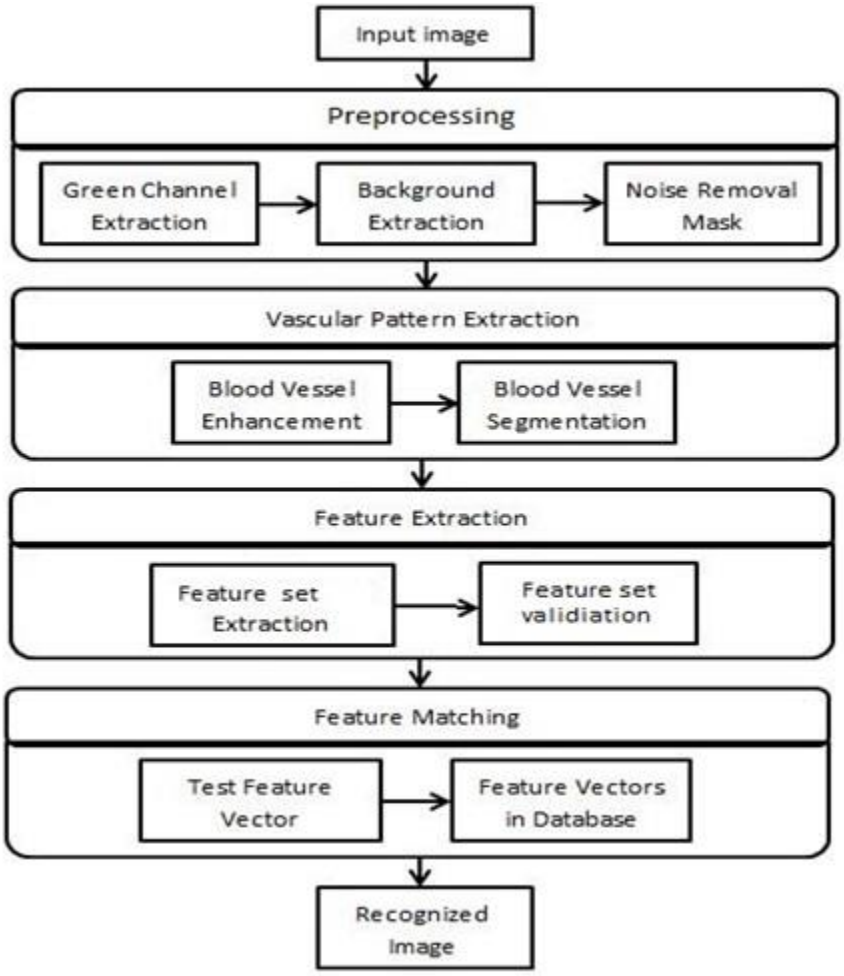
The following steps explain the process of iris scan: -

- Capturing an image for user iris by using camera with high resolution
- Extract the unique features of the iris
- The extracted features are stored into template
- During authentication, the scanned iris is compared to stored templates in the system.
- If the patterns match, the user is authenticated and granted access.

3. Retina scan

The retinal authentication system is mainly based on the unique blood vessel pattern in the retina, A retinal authentication system has basically three main steps: image acquisition, feature extraction, and feature matching

- The first stage of a retinal authentication system is image acquisition. To capture the image of retina a fundus camera is used, where the user must position his eye very close to the lens and the user must also remain perfectly still at this point..
- The second stage is feature extraction. Different features are extracted from the blood vessel structure of the retina image, and/or the retinal image itself. These features are unique for each individual and stored as a reference pattern.
- At the last stage, the features of test image are matched with the pre-stored template in the database depending on some criteria. If the criteria are satisfied then the person is authenticated.



4. Voice print

Voiceprint recognition was first proposed for the feasibility of human ear recognition mechanism and machine listening recognition. Voiceprint recognition technology can be divided into speaker recognition technology and speaker confirmation technology according to different functions, and can be divided into text correlation, text restriction and text independence according to different audio content.

The following steps explain the process of voice authentication:-

- ✓ **Voice Capture**: The user speaks into a microphone, and the system records their voice.

- ✓ **Feature Extraction**: The system analyzes unique voice characteristics like pitch, tone, rhythm, and frequency, converting them into a digital voiceprint (a mathematical model of the voice).
- ✓ **Template Storage**: During enrollment, the voiceprint is securely stored for future comparisons.
- ✓ **Voice Print Matching**: During authentication, the user's voice is captured again and converted into a new voiceprint, which is then compared to the stored voiceprint.
- ✓ **Decision**: If the voiceprints match, access is granted (authorization). If not, access is denied.

This process allows secure access based on the uniqueness of a individual's voice.

Authorization

Authorization is the part of access control concerned with restrictions on the actions of authenticated users. authorization deals with the situation where we've already authenticated Alice and we want to enforce restrictions on what she is allowed to do. Note that while authentication is binary (either a user is authenticated or not), authorization can be a much more fine-grained process.

Organizations often struggle to understand the difference between authentication and authorization. Authentication is the process of verifying individuals are who they say they are using biometric identification and MFA. The distributed nature of assets gives organizations many avenues for authenticating an individual.

Authorization is the act of giving individuals the correct data access based on their authenticated identity. One example of where authorization often falls short is if an individual leaves a job but still has access to that company's assets. This creates security holes

because the asset the individual used for work -- a smart phone with company software on it, for example -- is still connected to the company's internal infrastructure but is no longer monitored because the individual is no longer with the company. Left unchecked, this can cause major security problems for an organization. If the ex-employee's device were to be hacked, for example, the attacker could gain access to sensitive company data, change passwords or sell the employee's credentials or the company's data.One solution to this problem is strict monitoring and reporting on who has access to protected resources so, when a change occurs, it can be immediately identified and access control lists and permissions can be updated to reflect the change.

Access Control

Access control is a security technique that regulates who or what can view or use resources in a computing environment. It is a fundamental concept in security that minimizes risk to the business or organization.

In the context of network security, access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual.

There are two types of access control: physical and logical.

- ✓ Physical access control limits access to campuses, buildings, rooms and physical IT assets.
  Logical access control limits connections to computer networks, system files and data.

To secure a facility, organizations use electronic access control systems that rely on user credentials, access card readers, auditing and reports to track employee access to restricted business locations and proprietary areas, such as data centers.

Some of these systems incorporate access control panels to restrict entry to rooms and buildings, as well as alarms and lockdown capabilities, to prevent unauthorized access or operations.

- ✓ Logical access control systems **perform identification, authentication, and**

authorization of users and entities by evaluating required login credentials that can include passwords, personal identification numbers, biometric scans, security tokens or other authentication factors.

Multifactor authentication (MFA), which requires two or more authentication factors, is often an important part of a layered defense to protect access control systems

Why is access control important?

The goal of access control is to minimize the security risk of unauthorized access to physical and logical systems. Access control is a fundamental component of security compliance programs that ensures security technology and access control policies are in place to protect confidential information, such as customer data. Most organizations have infrastructure and procedures that limit access to networks, computer systems, applications, files and sensitive data, such as personally identifiable information and intellectual property.

Access control systems are complex and can be challenging to manage in dynamic IT environments that involve on-premises systems and cloud services. After high-profile breaches, technology vendors have shifted away from single sign-on systems to unified access management, which offers access controls for on- premises and cloud environments.

How access control works?

Access controls identify an individual or entity, verify the person or application is who or what it claims to be, and authorizes the access level and set of actions associated with the username or IP address. Directory services and protocols,including Lightweight Directory Access Protocol and Security Assertion Markup Language, provide access controls for authenticating and authorizing users and entities and enabling them to connect to computer resources, such as distributed applications and web servers. Organizations use different access control models depending on their compliance requirements and the security levels of IT they are trying to protect.

Access Control and Access Control tools

Access control is the process of:

- ✓ identifying a person doing a specific job
- ✓ authenticating them by looking at their identification
- ✓ granting a person only the key to the door or computer that they need accessto and nothing more.

In information security, one would look at this as:

- ✓ granting an individual permission to get onto a network via a username and password
- ✓ allowing them access to files, computers, or other hardware or software theyneed
- ✓ ensuring they have the right level of permission to do their job

So, how does one grant the right level of permission to an individual so that they can perform their duties? This is where access control models come into the picture.

Access Control Categories (Models)

Access control models have five types:

1. **Mandatory Access Control (MAC)**
2. **Role-Based Access Control (RBAC)**
3. **Discretionary Access Control (DAC)**
4. **Rule-Based Access Control (RBAC or RB-RBAC)**
5. **Attribute-based access control**

1. **Mandatory access control (MAC).** This is a security model in which access rights are regulated by a central authority based on multiple levels of security. Often used in government and military environments, classifications(e.g., confidential, secret, top secret) are assigned to system resources and the operating system or security kernel. MAC grants or denies access to resource objects based on the information security clearance of the user or device. For example, Security-Enhanced Linux is an implementation of MAC on Linux

2. **Discretionary access control (DAC).** This is an access control method in which owners or administrators of the protected system, data or resource set the policies defining who or what is authorized to access the resource. Many of these systems enable administrators to limit the propagation of access rights. A common criticism of DAC systems is a lack of centralized control. In a Discretionary Access Control (DAC) system, each resource, such as a file, directory, or program, is assigned an owner. Typically, the resource's creator is the owner, but ownership can be transferred or reassigned to another individual.
3. **Role-based access control (RBAC).** This is a widely used access control mechanism that restricts access to computer resources based on individuals or groups with defined business functions -- e.g., executive level, engineer level 1, etc. -- rather than the identities of individual users. The role-based security model relies on a complex structure of role assignments, role authorizations and role permissions developed using role engineering to regulate employee access to systems. RBAC systems can be used to enforce MAC and DAC frameworks.
4. **Rule-based access control.** This is a security model in which the system administrator defines the rules that govern access to resource objects. These rules are often based on conditions, such as time of day or location. It is not uncommonto use some form of both rule-based access control and RBAC to enforce access policies and procedures.
5. **Attribute-based access control.** This is a methodology that manages access rights by evaluating a set of rules, policies and relationships using the attributes of users(name,role,job..), systems and environmental conditions.

Implementing access control

Access control is integrated into an organization's IT environment. It can involve identity management and access management systems. These systems provide access control software, a user database and management tools for access control policies, auditing and enforcement.When a user is added to an access management system, system administrators use an automated provisioning system to set up permissions based on access control frameworks, job responsibilities and workflows

Challenges of access control

Many of the challenges of access control stem from the highly distributed nature of modern IT. It is difficult to keep track of constantly evolving assets because they are spread out both physically and logically. Specific examples of challenges include the following:

- dynamically managing distributed IT environments;
- password fatigue.
- compliance visibility through consistent reporting.
- centralizing user directories and avoiding application-specific silos; and
- data governance and visibility through consistent reporting.
- Another often overlooked challenge of access control is user experience. If an access management technology is difficult to use, employees may use it incorrectly or circumvent it entirely, creating security holes and compliance gaps.

Access control software

Many types of access control software and technology exist, and multiple components are often used together as part of a larger identity and access management (IAM) strategy. Software tools may be deployed on premises, in the cloud or both. They may focus primarily on a company's internal access management or outwardly on access management for customers. Types of access management software tools include the following:

- ✓reporting and monitoring applications
- ✓password management tools
- ✓provisioning tools
- ✓identity repositories
- ✓security policy enforcement tools

Microsoft Active Directory is one example of software that includes most of thetools listed above in a single offering. Other IAM vendors with popular products include IBM, Idaptive and Okta.

Access Control Matrix

The classic view of authorization begins with Lampson's access control matrix. This matrix contains all of the relevant information needed by an operating system to make decisions about which users are allowed to do what with the various system resources. We'll define a *subject* as a user of a system (not necessarily a human user) and an *object* as a system resource. Two fundamental constructs in thefield of authorization are *access control lists*, or ACLs, and *capabilities,* or C-lists. Both ACLs and C-lists are derived from Lampson's *access control matrix,* which hasa row for every subject and a column for every object. Sensibly enough, the access allowed by subject *S* to object *O* is stored at the intersection of the row indexed by *S*and the column indexed by *O*

An example of an access control matrix appears in Table 8.1, where we use UNIX-style notation, that is, x, r, and w stand for execute, read, and write privileges,respectively.

Table 8.1: Access Control Matrix

	OS	Accounting program	Accounting data	Insurance data	Payroll data
Bob	rx	rx	r	—	—
Alice	rx	rx	r	rw	rw
Sam	rwx	rwx	r	rw	rw
Accounting program	rx	rx	rw	rw	r

Notice that in Table 8.1, the accounting program is treated as both an object and a subject. This is a useful fiction, since we can enforce the restriction that the accounting data is only modified by the accounting program. As discussed in, the intent here is to make corruption of the accounting data more difficult, since any changes to the accounting data must be done by software that, presumably, includes standard accounting checks and balances. However, this does not prevent all possible attacks, since the system administrator, Sam, could replace the accounting program with a faulty (or fraudulent) version and thereby break the protection. But this trick does allow Alice and Bob to access the accounting data without allowing them to corrupt it—either intentionally or unintentionally.

1. ACLs and Capabilities

Since all subjects and all objects appear in the access control matrix, it contains all of the relevant information on which authorization decisions can be based. However, there is a practical issue in managing a large access control matrix.A system could have hundreds of subjects (or more) and tens of thousands of objects(or more), in which case an access control matrix with millions of entries (or more)would need to be consulted before any operation by any subject on any object. Dealing with such a large matrix could impose a significant burden on the system.

To obtain acceptable performance for authorization operations, the access control matrix can be partitioned into more manageable pieces. There are two obvious ways to split the access control matrix. First, we could split the matrix intoits columns and store each column with its corresponding object. Then, whenever anobject is accessed, its column of the access control matrix would be consulted to seewhether the operation is allowed. These columns are known as access control lists, or ACLs. For example, the ACL corresponding to insurance data in Table 8.1 is (Bob, —), (Alice, rw), (Sam, rw), (accounting program, rw).

Alternatively, we could store the access control matrix by row, where each row is stored with its corresponding subject. Then, whenever a subject tries to files is required. This illustrates one of the inherent advantages of capabilities perform anoperation, we can consult its row of the access control matrix to see if the operationis allowed. This approach is know as capabilities, or C-lists. For example, Alice's C-list in Table 8.1 is

(OS, rx), (accounting program, rx), (accounting data, r), (insurance data, rw),(payroll data, rw).

It might seem that ACLs and C-lists are equivalent, since they simply providedifferent ways of storing the same information. However, there are some subtle differences between the two approaches. Consider the comparison of ACLs and capabilities illustrated in Figure 8.1. Note that the arrows in Figure 8.1 point in opposite directions, that is, for ACLs, the arrows point from the resources to the users, while for capabilities, the arrows point from the users to the resources. This seemingly trivial difference has real significance. In particular, with capabilities, the association between users and files is built into the system, while for an ACL-based system, a separate method for associating users.

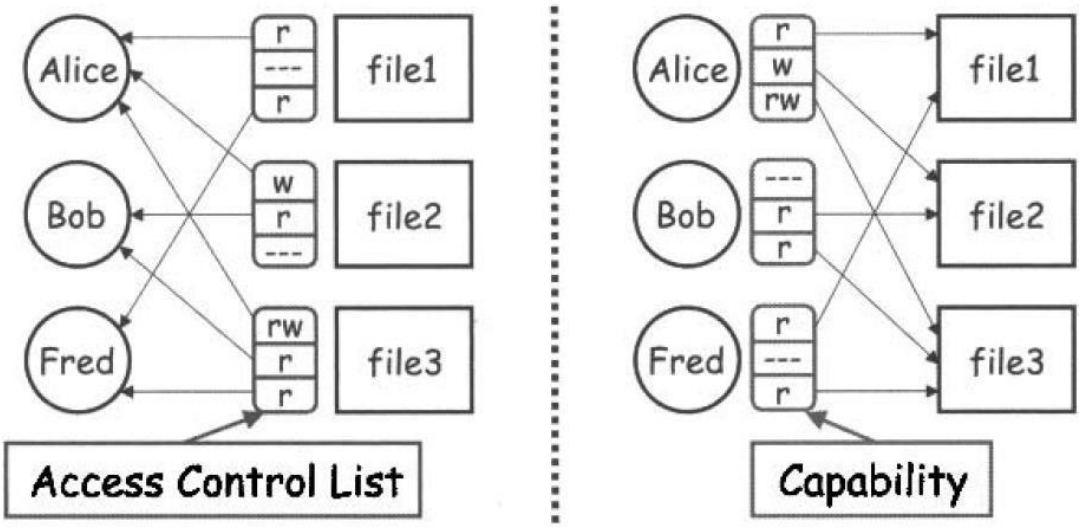


Figure 8.1: ACLs versus Capabilities

2. Confused Deputy

The *confused deputy* is a classic security problem that arises in many contexts. For our illustration of this problem, we consider a system with two resources, a compiler and a file named BILL that contains critical billing information, and one user, Alice. The compiler can write to any file, while Alicecan invoke the compiler and she can provide a filename where debugging information will be written. However, Alice is not allowed to write to the file BILL, since she might corrupt the billing information. The access control matrixfor this scenario appears in Table 8.2.

Table 8.2: Access Control Matrix for Confused Deputy Example

	Compiler	BILL
Alice	x	—
Compiler	rx	rw

Now suppose that Alice invokes the compiler, and she provides BILL as the debug filename. Alice does not have the privilege to access the file BILL, so this command should fail. However, the compiler, which is acting on Alice's behalf, doeshave the privilege to overwrite BILL. If the compiler acts with its privilege, then a side effect of Alice's command will be the trashing of the BILL file, as illustrated in Figure 8.2.

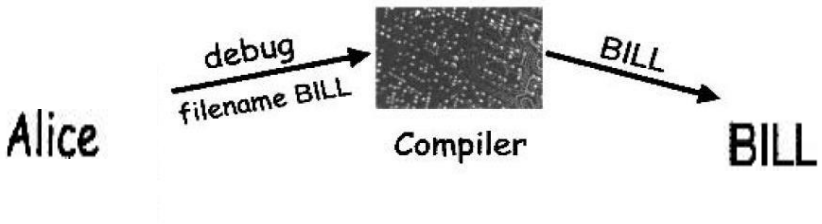


Figure 8.2: Confused Deputy

Why is this problem known as the confused deputy? The compiler is acting on Alice's behalf, so it is her deputy. The compiler is confused since it is acting basedon its own privileges when it should be acting based on Alice's privileges. With ACLs, it's more difficult (but not impossible) to avoid the confused deputy. In contrast, with capabilities it's relatively easy to prevent this problem, since capabilities are easily delegated, while ACLs are not. In a capabilities-based system,when Alice invokes the compiler, she can simply give her C-list to the compiler.

the compiler then consults Alice's C-list when checking privileges before attempting to create the debug file.

Since Alice does not have the privilege to overwrite BILL, the situation in Figure 8.2 can be avoided. A comparison of the relative advantages of ACLs and capabilities is instructive. ACLs are preferable when users manage their own files and when protection is data oriented. With ACLs, it's also easy to change rights to aparticular resource. On the other hand, with capabilities it's easy to delegate (and sub-delegate and sub-sub-delegate, and so on), and it's

easier to add or delete users.Due to the ability to delegate, it's easy to avoid the confused deputy when using capabilities. However, capabilities are more complex to implement and they have somewhat higher overhead—although it may not be obvious, many of the difficult issues inherent in distributed systems arise in the context of capabilities. For these reasons, ACLs are used in practice far more often than capabilities.

❖ Multilevel Security Models

In general, security models are descriptive, not proscriptive. That is, these models tell us what needs to be protected, but they don't answer the real question, that is, how to provide such protection. This is not a flaw in the models, as they are designed to set a framework for protection, but it is an inherent limitation on the practical utility of security modeling.

Multilevel security, or MLS, is familiar to all fans of spy novels, where classified information often figures prominently. In MLS, the subjects are the users (generally,human) and the objects are the data to be protected (for example, documents). Furthermore, *classifications* apply to objects while *clearances* apply to subjects. The

U.S. Department of Defense, or DoD, employs four levels of classifications and clearances, which can be ordered as

TOP SECRET > SECRET > CONFIDENTIAL > UNCLASSIFIED. (1)

For example, a subject with a SECRET clearance is allowed access to objects classified SECRET or lower but not to objects classified TOP SECRET. Apparently to make them more visible, security levels are generally rendered in upper case.

Let *O* be an object and *S* a subject. Then *O* has a classification and *S* has a clearance. The security *level* of *O* is denoted *L(0),* and the security level of *S* is similarly denoted *L(S).* In the DoD system, the four levels shown above in (1) are used for both clearances and classifications. Also, for a person to obtain a SECRETclearance, a more-or-less routine background check is required, while a TOP SECRET clearance requires an extensive background check, a polygraph exam, a psychological profile, etc.

Multilevel security is needed when subjects and objects at different levels use the same system resources. The purpose of an MLS system is to enforce a formof access control by restricting subjects so that they only access objects for which they have the necessary clearance. Military and government have long had an interest in MLS.

Today, there are many potential uses for MLS outside of its traditional classified government setting. For example, most businesses have information that is restricted to, say, senior management, and other information that is available to allmanagement, while still other proprietary information is available to everyone within the company and, finally, some information is available to everyone, including the general public. If this information is stored on a single system, the company must deal with MLS issues, even if they don't realize it. Note that these categories correspond directly to the

TOPSECRET, SECRET, CONFIDENTIAL, and UNCLASSIFIED classifications

There is also interest in MLS in such applications as network firewalls.

The goal in such a case is to keep an intruder, Trudy, at a low level to limit the damage that she can inflict after she breaches the firewall.

1. Bell-LaPadula model

The first security model that we'll consider is Bell-LaPadula, or BLP, which,believe it or not, was named after its inventors, Bell and LaPadula. The purpose of BLP is to capture the minimal requirements, with respect to confidentiality, that any MLS system must satisfy. BLP consists of the following **two statements:**

- **Simple Security Condition:** Subject *S* can read object *O* if and only if $L(0) < L(S)$.
- ***-Property** (Star Property): Subject *S* can write object *O* if and only if $L(S) < L(0)$.

The simple security condition merely states that Alice, for example, cannot read a document for which she lacks the appropriate clearance. This condition is clearly required of any MLS system.

The star property is somewhat less obvious. This property is designed to prevent, say, TOP

SECRET information from being written to, say, a SECRET document. This would break MLS security since a user with a SECRET clearance could then read TOP SECRET information. The writing could occur intentionally or, for example, as the result of a computer virus. In his groundbreaking work on viruses, Cohen mentions that viruses could be used to break MLS securit, and such attacks remain a very real threat to MLS systems today.

The simple security condition can be summarized as "no read up," while the star property implies "no write down." Consequently, BLP is sometimes succinctly stated as "no read up, no write down." It's difficult to imagine a security model that'sany simpler.

In response to McLean's criticisms, Bell and LaPadula fortified BLP with a *tranquility property.* Actually, there are two versions of this property. The strong tranquility property states that security labels can never change. This removes McLean's system Z from the BLP realm, but it's also impractical in the real world, since security labels must sometimes change. For example, the DoD regularly declassifies documents, which would be impossible under strict adherence to the strong tranquility property. For another example, it is often desirable to enforce *leastprivilege.* If a user has, say, a TOP SECRET clearance but is only browsing UNCLASSIFIED Web pages, it is desirable to only give the user an UNCLASSIFIED clearance, so as to avoid accidentally divulging classified information. If the user later needs a higher clearance, his active clearance can be upgraded. This is known as the *high water mark principle,* and we'll see it again when we discuss Biba's model, below.

Bell and Lapadula also offered a *weak tranquility property* in which a securitylabel can change, provided such a change does not violate an "established security policy." Weak tranquility can defeat system Z, and it can allow for least privilege, but the property is so vague as to be nearly meaningless for analytic purposes. Unfortunately, BLP may be too simple to be of any practical benefit.

2. Biba's Model

Whereas BLP deals with confidentiality, Biba's model deals with integrity. Infact, Biba's model is essentially an integrity version of BLP. If we trust the integrity of object *0*\ but not that of object O2, then if object *O* is composed of *0*\ and O2, we cannot trust the integrity of object *O*.

In other words, the integrity level of *O* is the minimum of the integrity of any object contained in *O*. Another way to say this is that for integrity, a low water mark principle holds. In contrast, for confidentiality, a high water mark principle applies. To state Biba's model formally, let *1(0)* denote the integrity of object *O* and *I(S)* the integrity of subject *S*. Biba's model is defined by the two statements:

- **Write Access Rule:** Subject *S* can write object *O* if and only if $1(0) < I(S)$.
- **Biba's Model:** A subject *S* can read the object *O* if and only if $I(S) < I(0)$.

The write access rule states that we don't trust anything that *S* writes any morethan we trust *S*. Biba's model states that we can't trust *S* any more than the lowest integrity object that *S* has read. In essence, we are concerned that *S* will be "contaminated" by lower integrity objects, so *S* is forbidden from viewing such objects.

Biba's model is actually very restrictive, since it prevents *S* from ever viewing an object at a lower integrity level.

Figure below illustrates the difference between BLP and Biba's model. Of course the fundamental difference is that BLP is for confidentiality, which implies a high water mark principle, while Biba is for integrity, which implies a low water mark principle.

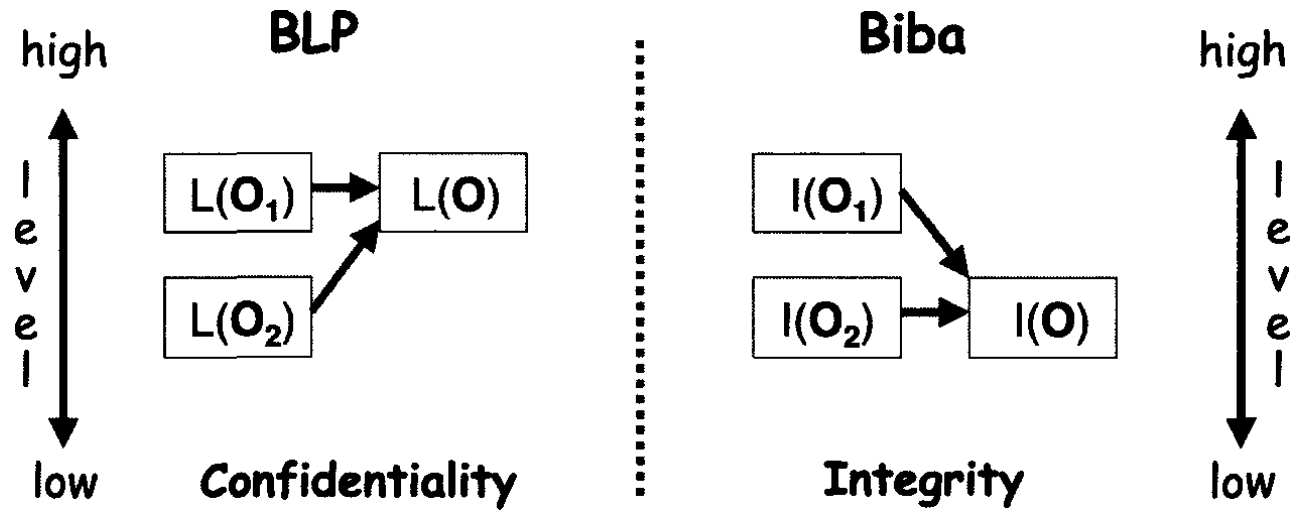


Figure 8.3: BLP versus Biba

❖ Compartments

Multilevel security systems enforce access control (or information flow) "up anddown," where the security levels are ordered in a hierarchy, such as (1). Usually, a simple hierarchy of security labels is not flexible enough to deal with a realistic situation. In practice, it is usually necessary to also use *compartments* to further restrict information flow "across" security levels. We use the notation SECURITY LEVEL {COMPARTMENT}
to denote a security level and its associated compartment or compartments. For example, suppose that we have compartments CAT and DOG within the TOP SECRET level. Then we would denote the resulting compartments as TOP SECRET {CAT} and TOP SECRET {DOG}. Note that there is also a TOP SECRET {CAT,DOG} compartment. While each of these compartments is TOP SECRET, a subject with a TOP SECRET clearance can only access a compartment if he or she is specifically allowed to do so. As a result, compartments have the effect of restricting information flow across security levels.
Compartments serve to enforce the ***need to know* principle**, that is, subjects are only allowed access to the information that they must know for their work. If a subject does not have a legitimate need to know everything at, say, the TOP SECRET level, then compartments can be used to limit the TOP SECRET information that the subject can access.

Why create compartments instead of simply creating a new classification level? It may be the case that, for example, TOP SECRET {CAT} and TOP SECRET {DOG} are not comparable, that is, neither

TOP SECRET {CAT} < TOP SECRET {DOG} nor TOP SECRET {CAT} > TOPSECRET {DOG}

holds. Using a strict MLS hierarchy, one of these two conditions must hold true. Consider the compartments in Figure 8.4, where the arrows represent ">" relationships. In this example, a subject with a TOP SECRET {CAT} clearance does not have access to information in the TOP SECRET {DOG} compartment. In addition, a subject with a TOP SECRET {CAT} clearance has access to the SECRET {CAT} compartment but not to the compartment SECRET {CAT,DOG}, even though the subject has a TOP SECRET clearance.Again, compartments provide a means to enforce the need to know principle.

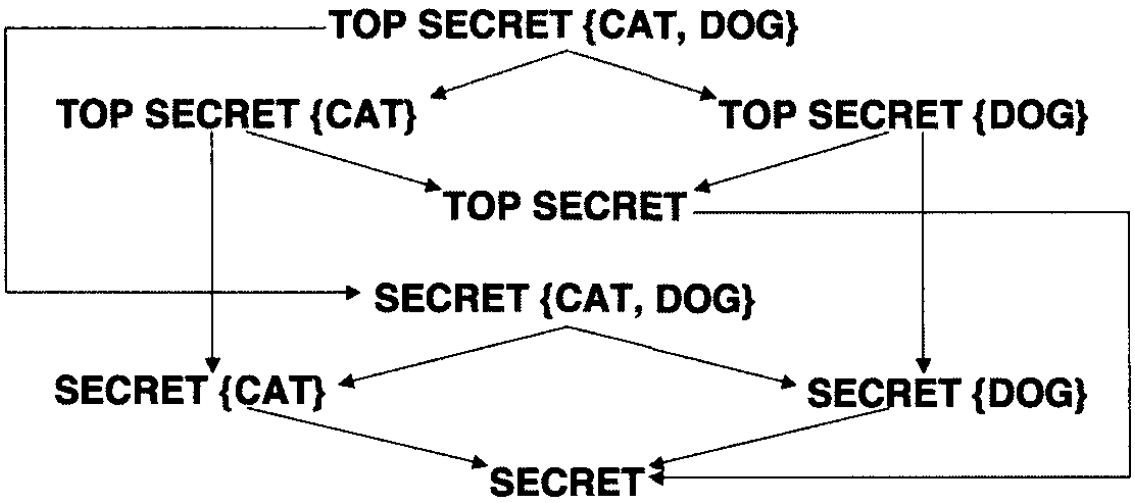


Figure 8.4: Compartments Example

❖ Covert Channel

A *covert channel* is a communication path not intended as such by the system's designers. Covert channels exist in many situations, but they are particularly prevalent in networks.

Covert channels are virtually impossible to eliminate, so the emphasis is instead on limiting the capacity of such channels. MLS systems are designed to restrict legitimate channels of communication. But a covert channel provides another way for information to flow. It is not difficult to give an example where resources shared by subjects at different security levels can be used to pass information, and thereby violate the security of an MLS system.

For example, suppose Alice has a TOP SECRET clearance while Bob only has a CONFIDENTIAL clearance. If the file space is shared by all users, then Aliceand Bob can agree that if Alice wants to send a 1 to Bob, she will create a file named,say, FileXYzW, and if she wants to send a 0 she will not create such a file. Bob cancheck to see whether file FileXYzW exists, and if it does, he knows Alice has sent him a 1, while if it does not, Alice has sent him a 0. In this way, a single bit of information has been passed through a covert channel, that is, through a means thatwas not intended for communication by the designers of the system. Note that Bob cannot look inside the file FileXYzW since he does not have the required clearance,but we are assuming that he can query the file system to see if such a file exists.

A single bit leaking from Alice to Bob is not a concern, but Alice could leak any amount of information by synchronizing with Bob. That is, Alice and Bob could agree that Bob will check for the file FileXYzW once each minute. As before, if the file does not exist, Alice has sent 0, and if it does exists, Alice has sent a 1. In this way Alice can (slowly) leak TOP SECRET information to Bob. This process is illustrated in Figure 8.5.

Covert channels are everywhere. For example, the print queue could be used to signal information in much the same way as in the example above. Networks area rich source of covert channels, and several hacking tools exist that exploit these covert channels.

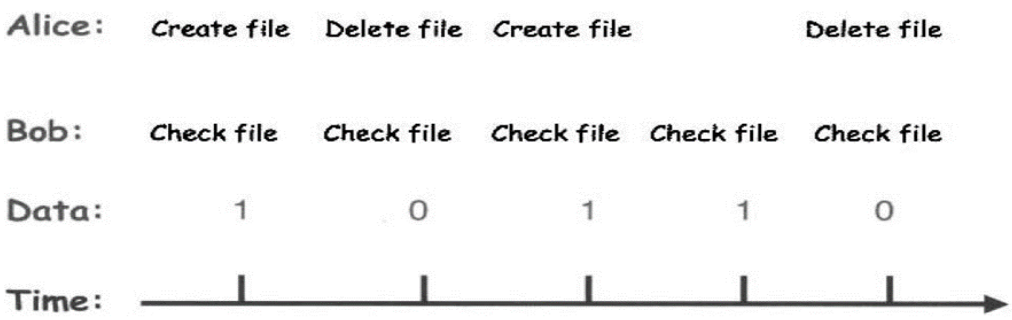


Figure 8.5: Covert Channel Example

Three things are required for a covert channel to exist.

- The sender and receiver must have access to a shared resource.
- The sender must be able to vary some property of the shared resource thatthe receiver can observe.
- The sender and receiver must be able to synchronize their communication.

From this description, it's apparent that potential covert channels really are everywhere. Of course, we can eliminate all covert channels— we just need to eliminate all shared resources and all communication. Obviously such a system would generally be of little use.

The conclusion here is that it's virtually impossible to eliminate all covert channels in any useful system. The DoD apparently agrees, since their guidelines merely call for reducing covert channel capacity to no more than one bit per second.The implication is that DoD has given up trying to eliminate covert channels.

Is a limit of one bit per second sufficient to prevent damage from covert channels? Consider a TOP SECRET file that is 100 MB in size. Suppose the plaintext version of this file is stored in a TOP SECRET file system, while an encrypted version of the file—encrypted with, say, AES using a 256-bit key—is stored in an UNCLASSIFIED location. Following the DoD guidelines, suppose thatwe have reduced the covert channel capacity of this system to 1 bit per second. Thenit would take more than 25 years to leak the entire 100 MB TOP SECRET documentthrough a covert channel. However, it would take less than 5 minutes to leak the 256-bit AES key through the same covert channel. The conclusion is that reducing covert channel capacity might be useful, but it will not be sufficient in all cases.

❖ Inference Control

Consider a database that includes information on college faculty in California. Suppose we query the database and ask for the average salary of female computer science professors at San Jose State University (SJSU) and we find the answer is

$100,000. We then query the database and ask for the number of female computer science professors at SJSU, and the answer is one. Then we could go to the SJSU computer science department website and determine the identity of this person.8 Inthis example, specific information has leaked from responses to general questions.

The goal of inference control is to prevent such leaks from happening, or at leastminimize the leakage. A database containing medical records would be of considerable interest to researchers. For example, by searching for statistical correlations, it may be possible to determine causes or risk factors for certain diseases. But patients want to keep their medical information private. How can we allow access to the statistically significant data while protecting privacy? An obviousfirst step is to remove names and addresses from the medical records. But this is notsufficient to ensure privacy as the college professor example above clearly demonstrates. What more can be done to provide stronger inference control while leaving the data accessible for legitimate research uses?

Several techniques used in inference control :

- ***query set size control***, in which no response is returned if the size of theset it too small. This approach would make it more difficult to determine the college professor's salary in the example above. However, if medical research is focused on a rare disease, query set sizecontrol could also prevent or distort important research.
- ***N-respondent, k% dominance rule,*** whereby data is not released if *k%*or more of the result is contributed by JV or fewer subjects. For example, we might query the census database and ask for the average net worth of individuals in Bill Gates' neighborhood. With any reasonable setting for *N* and *k* no results would be returned.
- ***randomization,*** that is, a small amount of random noise is added to the data. This is problematic in situations such as research into rare medicalconditions, where the noise might swamp legitimate data.

❖ CAPTCHA

The goal of CAPTCHA is to distinguish the human from the computer, based solely on the questions and answers. If the human questioner can't solve this puzzlewith a probability better than guessing, the computer passes the Turing test. This testis the gold standard in artificial intelligence, and no computer has yet passed the Turing test, but occasionally some claim to be getting close.

A "completely automated public Turing test to tell computers and humans apart," or *CAPTCHA,10* is a test that a human can pass, but a computer can't pass with a probability better than guessing [319]. This could be considered as a sort of inverseTuring test. The assumptions here are that the test is generated by a computer program and graded by a computer program, yet no computer can pass the test, evenif that computer has access to the source code used to generate the test. In other words, a "CAPTCHA is a program that can generate and grade tests that it itself cannot pass, much like some professors".

Since CAPTCHAs are designed to prevent non-humans from accessing resources, a CAPTCHA can be viewed as a form of access control. According to folklore, the original motivation for CAPTCHAs was an online poll that asked usersto vote for the best computer science graduate school.

The idea of a CAPTCHA to prevent automated "bots" from stuffing the ballot box. Today, CAPTCHAs are used in a wide variety of applications. For example, free email services use CAPTCHAs to prevent spammers from automatically signingup for large numbers of email accounts. The requirements for a CAPTCHA includethat it must be easy for most humans to pass and it must be difficult or impossible for a machines to pass, even if the machine has access to the CAPTCHA software. From the attacker's perspective, the only unknown is some randomness that is usedto generate the specific CAPTCHA. It is also desirable to have different types of CAPTCHAs in case some person cannot pass one particular type:

For example, many websites allow users to choose an audio CAPTCHA as an alternative to the usual visual CAPTCHA. An example of a CAPTCHA from [320]appears in Figure 8.7. In this case, a human might be asked to find three words thatappear in the image. This is a relatively

easy problem for humans and today it is alsoa fairly easy problem for computers to solve—much stronger CAPTCHAs exist.

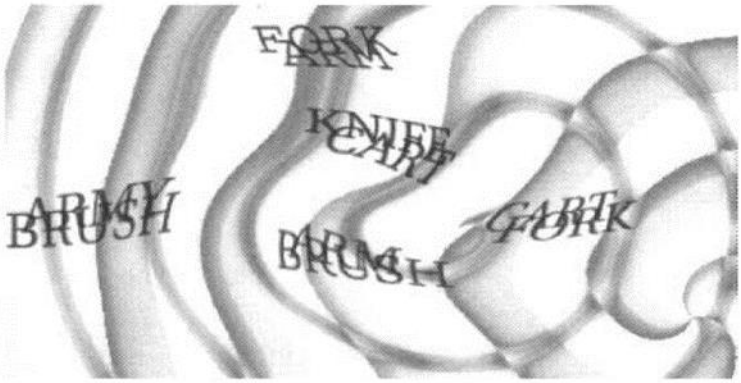


Figure 8.7: CAPTCHA (Courtesy of Luis von Ahn [320])

Sometimes computers are better than humans at solving all of the fundamentalvisual CAPTCHA problems, with one exception—the so-called segmentation problem, i.e., the problem of separating the letters from each other. Consequently, strong CAPTCHAs tend to

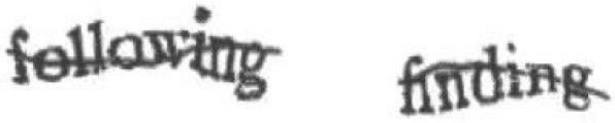


Figure 8.8: A Strong CAPTCHA [47]

References: -

1. Information security principles and practice, Mark Stamp, second edition,2011.
2. Cryptography and network security principles and practice, Fifth edition, William Stallings,2007.