Unit – I

Cryptography and Network Security

Definition of Computer Security:

The NIST Computer Security Handbook [NIST95] defines the term computer security as follows:

Computer Security: The protection afforded to an automated information system in order to attain the applicable objectives of preserving the integrity, availability, and confidentiality of information system resources (includes hardware, software, firmware, information/data, and telecommunications).

This definition introduces three key objectives that are at the heart of computer security:

Confidentiality: This term covers two related concepts:

Data confidentiality: Assures that private or confidential information is not made available or disclosed to unauthorized individuals.

Privacy: Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

• **Integrity:** This term covers two related concepts:

Data integrity: Assures that information and programs are changed only in a specified and authorized manner.

System integrity: Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

• **Availability:** Assures that systems work promptly and service is not denied to authorized users. These three concepts form what is often referred to as the CIA triad.

The three concepts embody the fundamental security objectives for both data and for information and computing services.

For example, the NIST standard FIPS 199 (Standards for Security Categorization of Federal Information and Information Systems) lists confidentiality, integrity, and availability as the three security objectives for information and for information systems.

FIPS 199 provides a useful characterization of these three objectives in terms of requirements and the definition of a loss of security in each category:

• **Confidentiality:** Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of

confidentiality is the unauthorized disclosure of information.

• **Integrity:** Guarding against improper information modification or destruction, including ensuring information nonrepudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information.

• **Availability:** Ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system

The Challenges of Computer Security Computer and network security is both fascinating and complex. Some of the reasons follow:

1. Security is not as simple as it might first appear to the novice. The requirements seem to be straightforward; indeed, most of the major requirements for security services can be given self-explanatory, one-word labels: confidentiality, authentication, nonrepudiation, or integrity. But the mechanisms used to meet those requirements can be quite complex, and understanding them may involve rather subtle reasoning.

2. In developing a particular security mechanism or algorithm, one must always consider potential attacks on those security features. In many cases, successful attacks are designed by looking at the problem in a completely different way, therefore exploiting an unexpected weakness in the mechanism.

3. Because of point 2, the procedures used to provide particular services are often counterintuitive. Typically, a security mechanism is complex, and it is not obvious from the statement of a particular requirement that such elaborate measures are needed. It is only when the various aspects of the threat are considered that elaborate security mechanisms make sense.

4. Having designed various security mechanisms, it is necessary to decide where to use them. This is true both in terms of physical placement (e.g., at what points in a network are certain security mechanisms needed) and in a logical sense (e.g., at what layer or layers of an architecture such as TCP/IP [Transmission Control Protocol/Internet Protocol] should mechanisms be placed).

5. Security mechanisms typically involve more than a particular algorithm or protocol. They also require that participants be in possession of some secret information (e.g., an encryption key), which raises questions about the creation, distribution, and protection of that secret information. There also may be a reliance on communications protocols whose behavior may complicate the task of developing the security mechanism.

6. Computer and network security is essentially a battle of wits between a peropetrator who tries to find holes and the designer or administrator who tries to close them. The great advantage that the attacker has is that he or she need only find a single weakness, while the designer must find and eliminate all weaknesses to achieve perfect security.

7. There is a natural tendency on the part of users and system managers to perceive little benefit from security investment until a security failure occurs.

8. Security requires regular, even constant, monitoring, and this is difficult in today's short-term, overloaded environment.

9. Security is still too often an afterthought to be incorporated into a system after the design is complete

rather than being an integral part of the design process.

10. Many users and even security administrators view strong security as an impediment to efficient and user-friendly operation of an information system or use of information

1.1. THE OSI SECURITY ARCHITECTURE

The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as follows:

- **Security attack:** Any action that compromises the security of information owned by an organization.
- **Security mechanism:** A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.
- **Security service:** A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization.
-

1.2. SECURITY ATTACKS

A useful means of classifying security attacks, is in terms of ***passive attacks*** and ***active attacks***. A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

PASSIVE ATTACKS

Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted.

Two types of passive attacks are **release of message contents** and **traffic analysis.**

The release of message contents is easily understood (Figure 1.1a). A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.

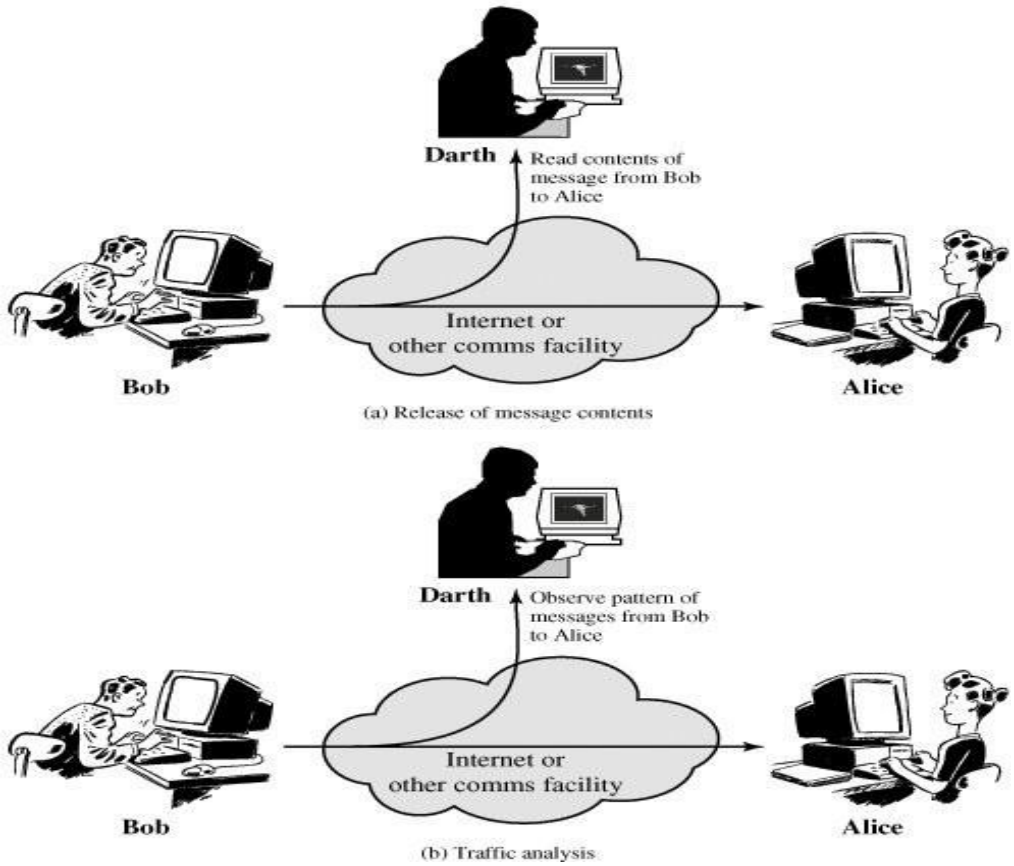


Figure 1.1. Passive Attacks

A second type of passive attack, traffic analysis, is subtler (Figure 1.1b). Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. Typically, the message traffic is not sent and received in an apparently normal fashion and the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern.

ACTIVE ATTACKS

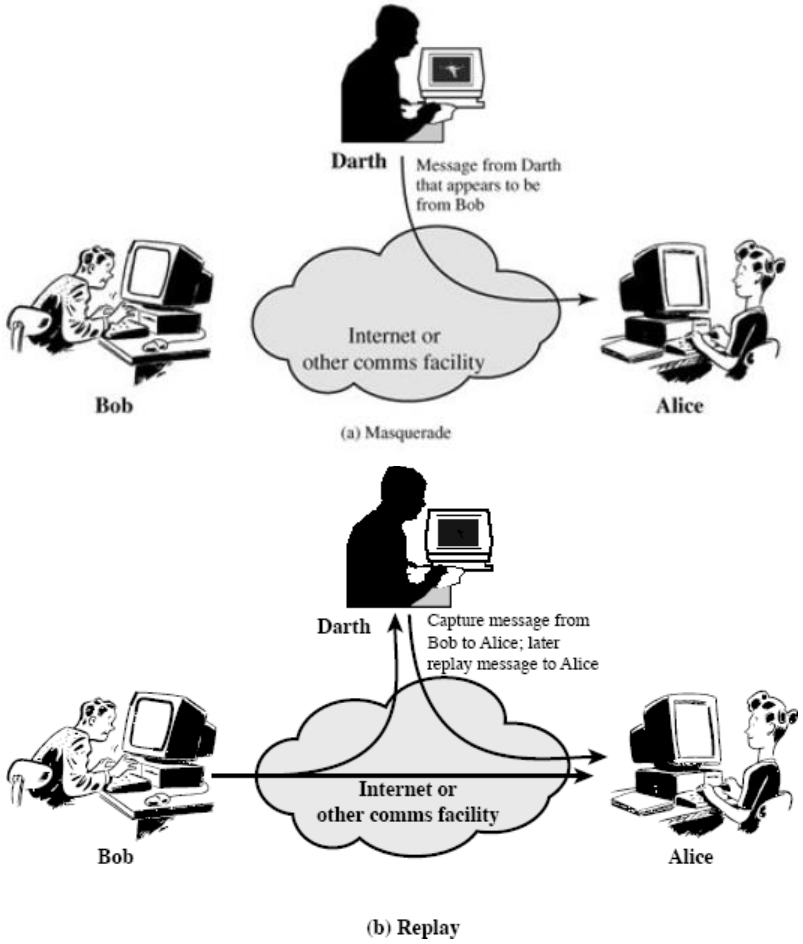
Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: **masquerade, replay, modification of messages, and denial of service.**

A **masquerade** takes place when one entity pretends to be a different entity (Figure 1.2a). A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.

Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect (Figure 1.2b).

Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect (Figure 1.2c). For example, a message meaning "Allow John Smith to read confidential file accounts" is modified to mean "Allow Fred Brown to read confidential file accounts."

The **denial of service** prevents or inhibits the normal use or management of communications facilities (Figure 1.2d). This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.



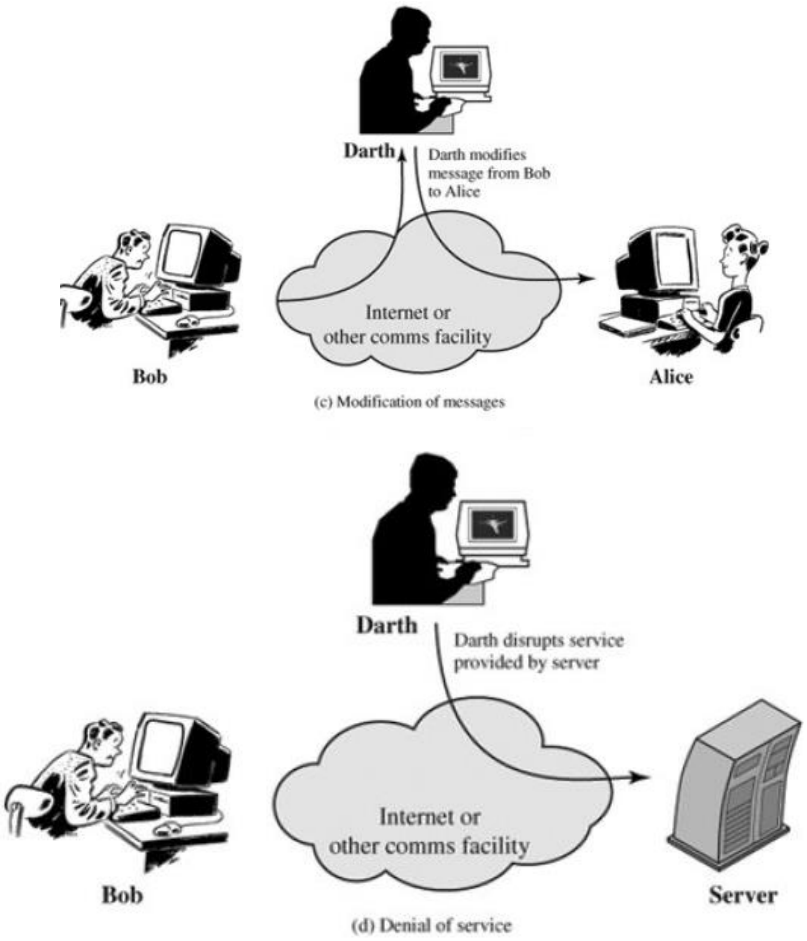


Figure 1.2. Active Attacks

1.3. SECURITY SERVICES

AUTHENTICATION

The assurance that the communicating entity is the one that it claims to be.

Peer Entity Authentication

Used in association with a logical connection to provide confidence in the identity of the entities connected.

Data Origin Authentication

In a connectionless transfer, provides assurance that the source of received data is as claimed.

ACCESS CONTROL

The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).

DATA CONFIDENTIALITY

The protection of data from unauthorized disclosure.

Connection Confidentiality

The protection of all user data on a connection.

Connectionless Confidentiality

The protection of all user data in a single data block

Selective-Field Confidentiality
The confidentiality of selected fields within the user data on a connection or in a single data block.
Traffic Flow Confidentiality
The protection of the information that might be derived from observation of traffic flows.
DATA INTEGRITY
The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).
Connection Integrity with Recovery
Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.
Connection Integrity without Recovery
As above, but provides only detection without recovery.
Selective-Field Connection Integrity
Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.
Connectionless Integrity
Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.
Selective-Field Connectionless Integrity
Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.
NONREPUDIATION
Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.
Nonrepudiation, Origin
Proof that the message was sent by the specified party.
Nonrepudiation, Destination
Proof that the message was received by the specified party.
AUTHENTICATION
The authentication service is concerned with assuring that a communication is authentic. In the case of a single message, such as a warning or alarm signal, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an ongoing interaction, such as the connection of a terminal to a host, two aspects are involved. First, at the time of connection initiation, the service assures that the two entities are authentic, that is, that each is the entity that it claims to be. Second, the service must assure that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties for the purposes of unauthorized transmission or reception.
Two specific authentication services are defined in X.800:
Peer entity authentication: Provides for the corroboration of the identity of a peer entity in an association. It is provided for use at the establishment of, or at times during the data transfer phase of, a connection. It attempts to provide confidence that an entity is not performing either a masquerade or an unauthorized replay of a previous connection.
Data origin authentication: Provides for the corroboration of the source of a data unit. It does

not provide protection against the duplication or modification of data units. This type of service supports applications like electronic mail where there are no prior interactions between the communicating entities.

ACCESS CONTROL

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.

DATA CONFIDENTIALITY

Confidentiality is the protection of transmitted data from passive attacks. With respect to the content of a data transmission, several levels of protection can be identified. The broadest service protects all user data transmitted between two users over a period of time. For example, when a TCP connection is set up between two systems, this broad protection prevents the release of any user data transmitted over the TCP connection. Narrower forms of this service can also be defined, including the protection of a single message or even specific fields within a message. These refinements are less useful than the broad approach and may even be more complex and expensive to implement. The other aspect of confidentiality is the protection of traffic flow from analysis. This requires that an attacker not be able to observe the source and destination, frequency, length, or other characteristics of the traffic on a communications facility.

DATA INTEGRITY

As with confidentiality, integrity can apply to a stream of messages, a single message, or selected fields within a message. Again, the most useful and straightforward approach is total stream protection.

A connection-oriented integrity service, one that deals with a stream of messages, assures that messages are received as sent, with no duplication, insertion, modification, reordering, or replays. The destruction of data is also covered under this service. Thus, the connection-oriented integrity service addresses both message stream modification and denial of service. On the other hand, a connectionless integrity service, one that deals with individual messages without regard to any larger context, ge nerallyprovides protection against message modification only.

We can make a distinction between the service with and without recovery. Because the integrity service relates to active attacks, we are concerned with detection rather than prevention. If a violation of integrity is detected, then the service may simply report this violation, and some other portion of software or human intervention is required to recover from the violation. Alternatively, there are mechanisms available to recover from the loss of integrity of data, as we will review subsequently. The incorporation of automated recovery mechanisms is, in general, the more attractive alternative.

NONREPUDIATION

Nonrepudiation prevents either sender or receiver from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the alleged sender in fact sent the message. Similarly, when a message is received, the sender can prove that the alleged receiver in fact received the message.

1.4. SECURITY MECHANISMS

There are two types of security Mechanisms 1. Specific Security Mechanisms 2. Pervasive Security Mechanisms

SPECIFIC SECURITY MECHANISMS

May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

Encipherment

The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.

Digital Signature

Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).

Access Control

A variety of mechanisms that enforce access rights to resources.

Data Integrity

A variety of mechanisms used to assure the integrity of a data unit or stream of data units.

Authentication Exchange

A mechanism intended to ensure the identity of an entity by means of information exchange.

Traffic Padding

The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.

Routing Control

Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.

Notarization

The use of a trusted third party to assure certain properties of a data exchange.

PERVASIVE SECURITY MECHANISMS

Mechanisms that are not specific to any particular OSI security service or protocol layer.

Trusted Functionality

That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).

Security Label

The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.

Event Detection

Detection of security-relevant events.

Security Audit Trail

Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.

Security Recovery

Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

1.5. A Model for Network Security

A model for much of what we will be discussing is captured, in very general terms, in Figure 1.3. A message is to be transferred from one party to another across some sort of internet. The two parties, who are the principals in this transaction, must cooperate for the exchange to take place. A logical information channel is established by defining a route through the internet from source

to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.

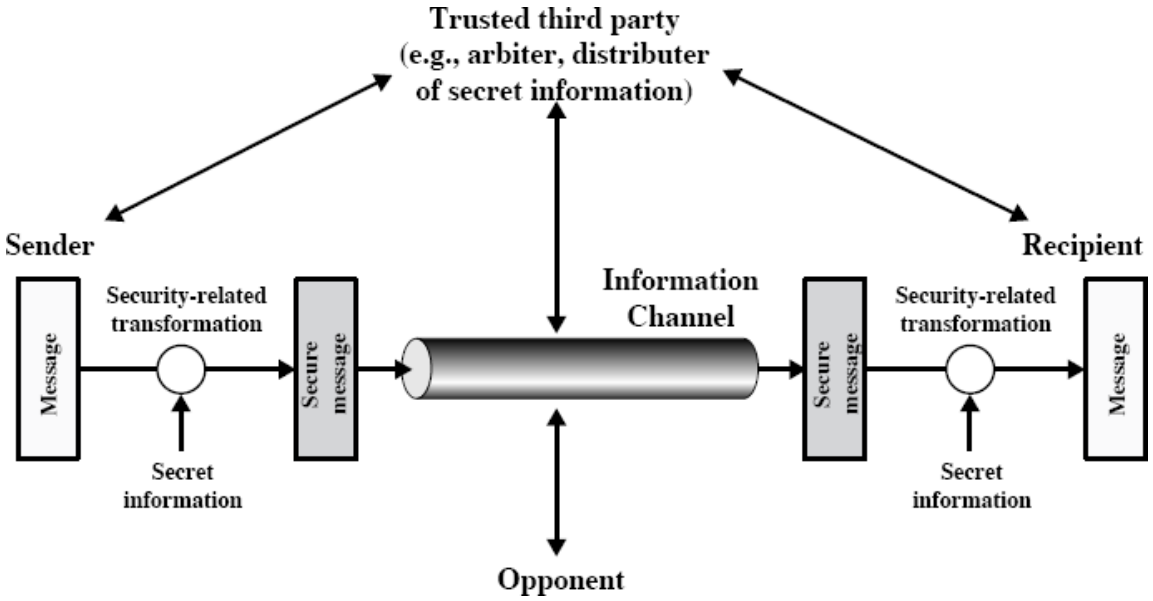


Figure 1.3. Model for Network Security

Security aspects come into play when it is necessary or desirable to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity, and so on. All the techniques for providing security have two components:

- A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender
- Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission.

This general model shows that there are four basic tasks in designing a particular security service:

- Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.
- Generate the secret information to be used with the algorithm.
- Develop methods for the distribution and sharing of the secret information.
- Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

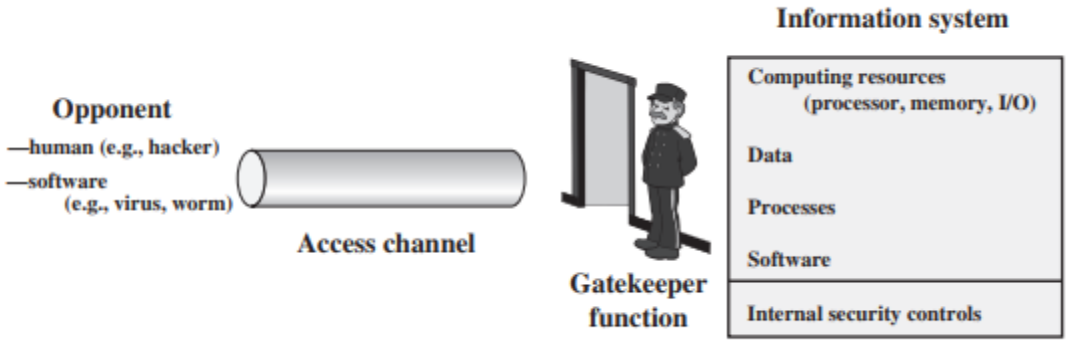


Figure 1.3 Network Access Security Model

• Information access threats: Intercept or modify data on behalf of users who should not have access to that data.
• Service threats: Exploit service flaws in computers to inhibit use by legitimate users
Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a disk that contains the unwanted logic concealed in otherwise useful software. They can also be inserted into a system across a network; this latter mechanism is of more concern in network security.
The security mechanisms needed to cope with unwanted access fall into two broad categories (see Figure 1.3). The first category might be termed a gatekeeper function. It includes password-based login procedures that are designed to deny access to all but authorized users and screening logic that is designed to detect and reject worms, viruses, and other similar attacks. Once either an unwanted user or unwanted software gains access, the second line of defense consists of a variety of internal controls that monitor activity and analyze stored information in an attempt to detect the presence of unwanted intruders.

1.6. CLASSICAL ENCRYPTION TECHNIQUES

- Symmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the same key. It is also known as conventional encryption.
- Symmetric encryption transforms plaintext into ciphertext using a secret key and an encryption algorithm. Using the same key and a decryption algorithm, the plaintext is recovered from the ciphertext.
- The two types of attack on an encryption algorithm are cryptanalysis, based on properties of the encryption algorithm, and brute-force, which involves trying allpossible keys.
- Traditional (precomputer) symmetric ciphers use substitution and/or transposition techniques. Substitution techniques map plaintext elements (characters, bits) into ciphertext elements. Transposition techniques systematically transpose the positions of plaintext elements.
- Rotor machines are sophisticated precomputer hardware devices that use substitution techniques.
- Steganography is a technique for hiding a secret message within a larger one in such a way that others cannot discern the presence or contents of the hidden message.

Before beginning, we define some terms. An original message is known as the **plaintext**, while the coded message is called the **ciphertext**. The process of converting from plaintext to ciphertext is known as **enciphering or encryption**; restoring the plaintext from the ciphertext is **deciphering or decryption**. The many schemes used for encryption constitute the area of study known as **cryptography**. Such a scheme is known as a **cryptographic system or a cipher.** Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of **cryptanalysis**. Cryptanalysis is what the layperson calls "breaking the code." The areas of cryptography and cryptanalysis together are called **cryptology**.

1.7. SYMMETRIC CIPHER MODEL

A symmetric encryption scheme has five ingredients (Figure 1.4):

- **Plaintext:** This is the original intelligible message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various substitutions and
- transformations on the plaintext.
- **Secret key:** The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.
- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

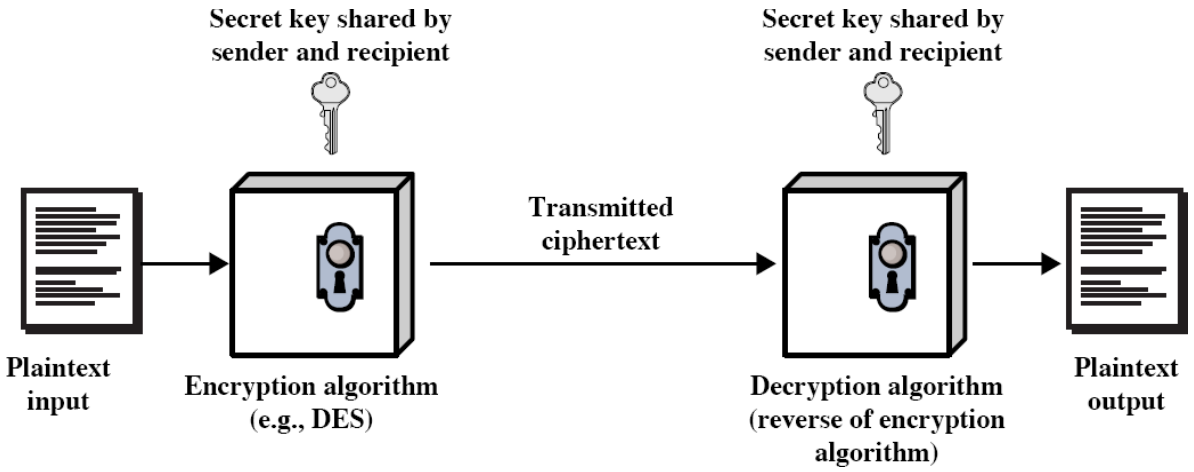


Figure 1.4. Simplified Model of Conventional Encryption

There are two requirements for secure use of conventional encryption:

We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. This requirement is usually stated in a stronger form: The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext

Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

We assume that it is impractical to decrypt a message on the basis of the ciphertext plus knowledge of the encryption/decryption algorithm. In other words, we do not need to keep the algorithm secret; we need to keep only the key secret.

Let us take a closer look at the essential elements of a symmetric encryption scheme, using Figure 1.5 .A source produces a message in plaintext, X = [X1, X2, ..., XM]. The M elements of X are letters in some finite alphabet. Traditionally, the alphabet usually consisted of the 26 capital letters. Nowadays, the binary alphabet {0, 1} is typically used. For encryption, a key of the form K = [K1, K2, ..., KJ] is generated. If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel. Alternatively, a third party could generate the key and securely deliver it to both source and destination.

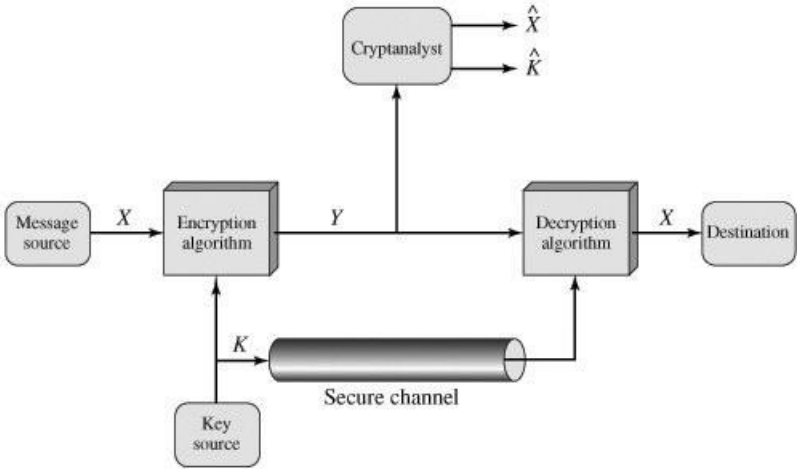


Figure 1.5 Model of Conventional Cryptosystem

With the message X and the encryption key K as input, the encryption algorithm forms the ciphertext Y = [Y1, Y2, ..., YN]. We can write this as Y = E(K, X) This notation indicates that Y is produced by using encryption algorithm E as a function of the plaintext X, with the specific function determined by the value of the key K.

The intended receiver, in possession of the key, is able to invert the transformation:

X = D(K, Y)

An opponent, observing Y but not having access to K or X, may attempt to recover X or K or both X and K. It is assumed that the opponent knows the encryption (E) and decryption (D) algorithms. If the opponent is interested in only this particular message, then the focus of the effort is to recover X by generating a plaintext estimate $\hat{X}$. Often, however, the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover K by generating an estimate $\hat{K}$.

Cryptography

Cryptographic systems are characterized along three independent dimensions:

- **The type of operations used for transforming plaintext to ciphertext.** All encryption algorithms are based on two general principles: substitution, in which each element in the
- plaintext (bit, letter, group of bits or letters) is mapped into another element, and transposition, in which elements in the plaintext are rearranged. The fundamental requirement is that no information be lost (that is, that all operations are reversible). Most systems, referred to as product systems, involve multiple stages of substitutions and transpositions.
- 2. **The number of keys used**. If both sender and receiver use the same key, the system is
- referred to as symmetric, single-key, secret-key, or conventional encryption. If the sender and receiver use different keys, the system is referred to as asymmetric, two-key, or public-key encryption.
- 3. **The way in which the plaintext is processed**. A block cipher processes the input one block of elements at a time, producing an output block for each input block. A stream cipher processes the input elements continuously, producing output one element at a time, as it goes along.

Cryptanalysis

Typically, the objective of attacking an encryption system is to recover the key in use rather then simply to recover the plaintext of a single ciphertext. There are two general approaches to attacking a conventional encryption scheme:

- **Cryptanalysis:** Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.
- **Brute-force attack:** The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

1.8. SUBSTITUTION TECHNIQUES

a.Caesar Cipher

The earliest known use of a substitution cipher, and the simplest, was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing three places further down the alphabet. For example,

plain: meet me after the toga party
cipher: PHHW PH DIWHU WKH WRJD SDUWB

Note that the alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows:

plain: a b c d e f g h i j k l m n o p q r s t u v w x y z
cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Let us assign a numerical equivalent to each letter:

a	B	C	d	e	f	g	h	i	j	K	L	m
0	1	2	3	4	5	6	7	8	9	10	11	12

n	O	P	q	r	s	t	u	v	w	X	y	z
13	14	15	16	17	18	19	20	21	22	23	24	25

Then the algorithm can be expressed as follows. For each plaintext letter p, substitute the ciphertext letter C

$$C = E(3, p) = (p + 3) \bmod 26$$

A shift may be of any amount, so that the general Caesar algorithm is

$$C = E(k, p) = (p + k) \bmod 26$$

where k takes on a value in the range 1 to 25. The decryption algorithm is simply

$$p = D(k, C) = (C\ k) \bmod 26$$

If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis is easily performed: Simply try all the 25 possible keys.

Three important characteristics of this problem enabled us to use a brute-force cryptanalysis:

- The encryption and decryption algorithms are known.
- There are only 25 keys to try.
- The language of the plaintext is known and easily recognizable.

b.Playfair Cipher

The best-known multiple-letter encryption cipher is the Playfair, which treats digrams in the plaintext as single units and translates these units into ciphertext digrams. The Playfair algorithm is based on the use of a 5 x 5 matrix of letters constructed using a keyword.

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	O	S	T
U	V	W	X	Z

In this case, the keyword is **monarchy**. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetic order. The letters I and J count as one letter.

Plaintext is encrypted two letters at a time, according to the following rules:

- Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.
- Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, ar is encrypted as RM.
- Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, mu is encrypted as CM.
- Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

c.Hill Cipher

Another interesting multiletter cipher is the Hill cipher, developed by the mathematician Lester Hill in 1929. The encryption algorithm takes m successive plaintext letters and substitutes for them m ciphertext letters. The substitution is determined by m linear equations in which each character is assigned a numerical value (a = 0, b = 1 ... z = 25). For m = 3, the system can be described as follows:

$$c1 = (k11P1 + k12P2 + k13P3) \bmod 26$$
$$c2 = (k21P1 + k22P2 + k23P3) \bmod 26$$
$$c3 = (k31P1 + k32P2 + k33P3) \bmod 26$$

This can be expressed in term of column vectors and matrices:

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \bmod 26$$

or

$$C = KP \bmod 26$$

where C and P are column vectors of length 3, representing the plaintext and ciphertext, and K is a 3 x 3 matrix, representing the encryption key. Operations are performed mod 26.

For example, consider the plaintext "paymoremoney" and use the encryption key

$$\mathbf{K} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

The first three letters of the plaintext are represented by the vector $\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix}$. Then $\mathbf{K}\begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix} = \begin{pmatrix} 375 \\ 819 \\ 486 \end{pmatrix} \bmod 26 = \begin{pmatrix} 11 \\ 13 \\ 18 \end{pmatrix}$ = LNS. Continuing in this fashion, the ciphertext for the entire plaintext is LNSHDLEWMTRW.

Decryption requires using the inverse of the matrix K. The inverse K1 of a matrix K is defined by the equation $KK^{-1} = K^{-1}K = I$, where I is the matrix that is all zeros except for ones along the main diagonal from upper left to lower right. The inverse of a matrix does not always exist, but when it does, it satisfies the preceding equation. In this case, the inverse is:

$$\mathbf{K}^{-1} = \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix}$$

If key is 2 × 2 matrix , take two plain text at a time

Suppose that the plaintext "friday" is encrypted using a 2 x 2 Hill cipher to yield the ciphertext PQCFKU. Thus, we know that

$$\mathbf{K}\begin{pmatrix} 5 \\ 17 \end{pmatrix} \bmod 26 = \begin{pmatrix} 15 \\ 16 \end{pmatrix}; \mathbf{K}\begin{pmatrix} 8 \\ 3 \end{pmatrix} \bmod 26 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}; \quad \text{and} \quad \mathbf{K}\begin{pmatrix} 0 \\ 24 \end{pmatrix} \bmod 26 = \begin{pmatrix} 10 \\ 20 \end{pmatrix}$$

Using the first two plaintext-ciphertext pairs, we have

$$\begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix} = \mathbf{K}\begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} \bmod 26$$

The inverse of X can be computed:

$$\begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix}^{-1} = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

so

$$\mathbf{K} = \begin{pmatrix} 15 & 2 \\ 16 & 5 \end{pmatrix}\begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix} = \begin{pmatrix} 137 & 60 \\ 149 & 107 \end{pmatrix} \bmod 26 = \begin{pmatrix} 7 & 8 \\ 19 & 3 \end{pmatrix}$$

This result is verified by testing the remaining plaintext-ciphertext pair.

d.Polyalphabetic Ciphers:

Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is polyalphabetic substitution cipher.

All these techniques have the following features in common:

- A set of related monoalphabetic substitution rules is used.
- A key determines which particular rule is chosen for a given transformation

To aid in understanding the scheme and to aid in its use, a matrix known as the Vigenère tableau is constructed.(Figure 1.6). Each of the 26 ciphers is laid out horizontally, with the key letter for each cipher to its left. A normal alphabet for the plaintext runs across the top. The process of

encryption is simple: Given a key letter x and a plaintext letter y, the ciphertext letter is at the intersection of the row labeled x and the column labeled y; in this case the ciphertext is V.

		Plaintext																								
		a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Key	a	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	b	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
	c	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
	d	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
	e	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
	f	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
	g	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
	h	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
	i	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
	j	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
	k	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
	l	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
	m	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
	n	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
	o	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
	p	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
	q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
	r	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
	s	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
	t	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
	u	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
	v	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
	w	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
	x	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
	z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure 1.6 . The Modern Vigenère Tableau

To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword. For example, if the keyword is deceptive, the message "we are discovered save yourself" is encrypted as follows:

key: deceptivedeceptivedeceptive
plaintext: wearediscoveredsaveyourself
ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Decryption is equally simple. The key letter again identifies the row. The position of the ciphertext letter in that row determines the column, and the plaintext letter is at the top of that column.

1.9 TRANSPOSITION TECHNIQUES

All the techniques examined so far involve the substitution of a ciphertext symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

The simplest such cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. For example, to encipher the message "meet me after the toga party" with a rail fence of depth 2, we write the following:

```
m e m a t r h t g p r y
 e t e f e t e o a a t
```

The encrypted message is

MEMATRHTGPRYETEFETEOAAT

This sort of thing would be trivial to cryptanalyze. A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute

the order of the columns. The order of the columns then becomes the key to the algorithm. For example,

Key: 4 3 1 2 5 6 7
Plaintext: a t t a c k p
o s t p o n e
d u n t i l t
w o a m x y z
Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. For the type of columnar transposition just shown, cryptanalysis is fairly straightforward and involves laying out the ciphertext in a matrix and playing around with column positions. Digram and trigram frequency tables can be useful.

The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is a more complex permutation that is not easily reconstructed. Thus, if the foregoing message is reencrypted using the same algorithm,

Key: 4 3 1 2 5 6 7
Input: t t n a a p t
m t s u o a o
d w c o i x k
n l y p e t z
Output: NSCYAUOPTTWLTMDNAOIEPAXTTOKZ

1.10. ROTOR MACHINES

The basic principle of the rotor machine is illustrated in Figure 1.7. The machine consists of a set of independently rotating cylinders through which electrical pulses can flow. Each cylinder has 26 input pins and 26 output pins, with internal wiring that connects each input pin to a unique output pin. For simplicity, only three of the internal connections in each cylinder are shown.

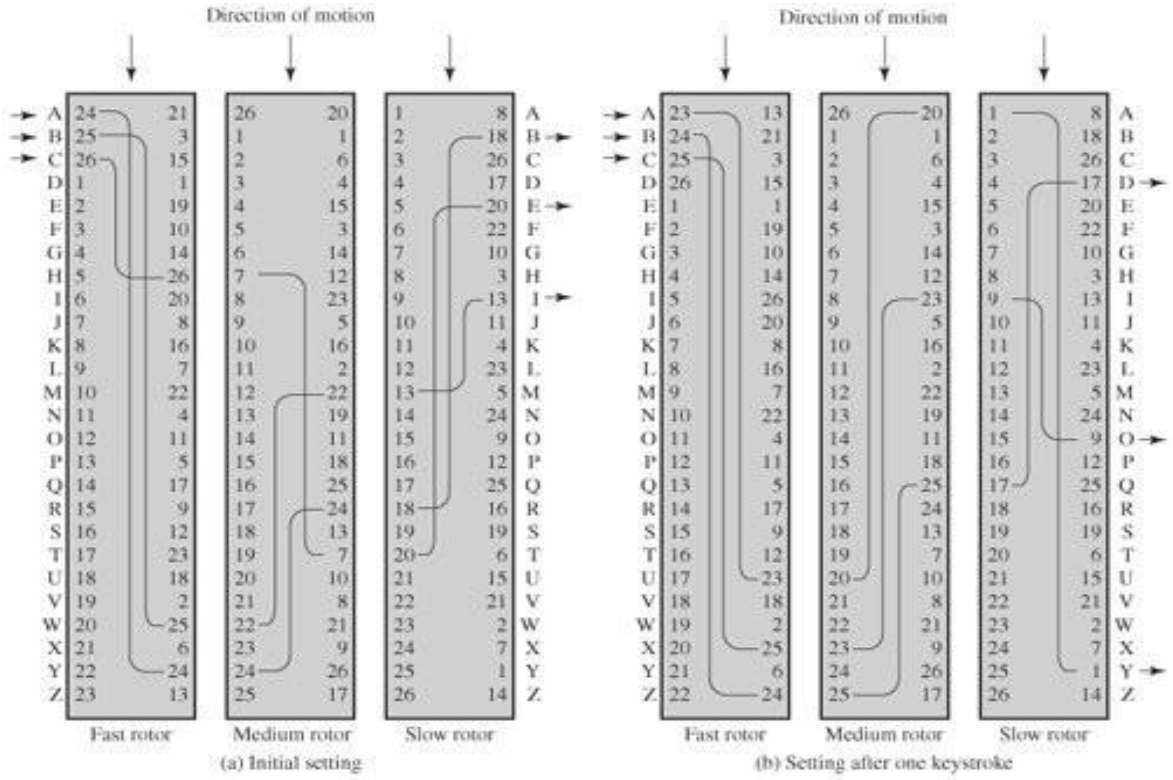


Figure 1.7. Three-Rotor Machine with Wiring Represented by Numbered Contacts

If we associate each input and output pin with a letter of the alphabet, then a single cylinder defines a monoalphabetic substitution. For example, in Figure 1.7, if an operator depresses the key for the letter A, an electric signal is applied to the first pin of the first cylinder and flows through the internal connection to the twenty-fifth output pin.

Consider a machine with a single cylinder. After each input key is depressed, the cylinder rotates one position, so that the internal connections are shifted accordingly. Thus, a different monoalphabetic substitution cipher is defined. After 26 letters of plaintext, the cylinder would be back to the initial position. Thus, we have a polyalphabetic substitution algorithm with a period of 26.

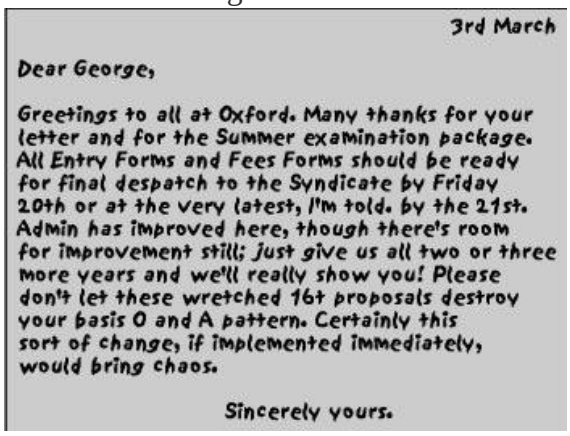
A single-cylinder system is trivial and does not present a formidable cryptanalytic task. The power of the rotor machine is in the use of multiple cylinders, in which the output pins of one cylinder are connected to the input pins of the next. Figure 1.7 shows a three-cylinder system. The left half of the figure shows a position in which the input from the operator to the first pin (plaintext letter a) is routed through the three cylinders to appear at the output of the second pin (ciphertext letter B).

With multiple cylinders, the one closest to the operator input rotates one pin position with each Key stroke. The right half of Figure 1.7 shows the system's configuration after a single keystroke. For every complete rotation of the inner cylinder, the middle cylinder rotates one pin position. Finally, for every complete rotation of the middle cylinder, the outer cylinder rotates one pin position. This is the same type of operation seen with an odometer. The result is that there are 26 x 26 x 26 = 17,576 different substitution alphabets used before the system repeats. The addition of fourth and fifth rotors results in periods of 456,976 and 11,881,376 letters, respectively.

1.11. STEGANOGRAPHY

A plaintext message may be hidden in one of two ways. The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text

A simple form of steganography, but one that is time-consuming to construct, is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message. For example, the sequence of first letters of each word of the overall message spells out the hidden message. Figure 1.8 shows an example in which a subset of the words of the overall message is used to convey the hidden message.



3rd March

Dear George,

Greetings to all at Oxford. Many thanks for your letter and for the Summer examination package. All Entry Forms and Fees Forms should be ready for final despatch to the Syndicate by Friday 20th or at the very latest, I'm told. by the 21st. Admin has improved here, though there's room for improvement still; just give us all two or three more years and we'll really show you! Please don't let these wretched 16t proposals destroy your basis O and A pattern. Certainly this sort of change, if implemented immediately, would bring chaos.

Sincerely yours.

Figure 1.8. A Puzzle for Inspector Morse

Various other techniques have been used historically; some examples are the following

- **Character marking:** Selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held at an angle to bright light.
- **Invisible ink:** A number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.
- **Pin punctures:** Small pin punctures on selected letters are ordinarily not visible unless the paper is held up in front of a light.
- **Typewriter correction ribbon:** Used between lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Although these techniques may seem archaic, they have contemporary equivalents. [WAYN93] proposes hiding a message by using the least significant bits of frames on a CD. For example, the Kodak Photo CD format's maximum resolution is 2048 by 3072 pixels, with each pixel containing 24 bits of RGB color information. The least significant bit of each 24-bit pixel can be changed without greatly affecting the quality of the image. The result is that you can hide a 2.3-megabyte message in a single digital snapshot. There are now a number of software packages available that take this type of approach to steganography.

Steganography has a number of **drawbacks** when compared to encryption. It requires a lot of overhead to hide a relatively few bits of information, although using some scheme like that proposed in the preceding paragraph may make it more effective. Also, once the system is discovered, it becomes virtually worthless. This problem, too, can be overcome if the insertion method depends on some sort of key . Alternatively, a message can be first encrypted and then hidden using steganography.

The **advantage** of steganography is that it can be employed by parties who have something to

lose should the fact of their secret communication (not necessarily the content) be discovered. Encryption flags traffic as important or secret or may identify the sender or receiver as someone with something to hide.

Reference Book :

Cryptography and Network Security Principles and Practices, Fourth Edition,By William Stallings

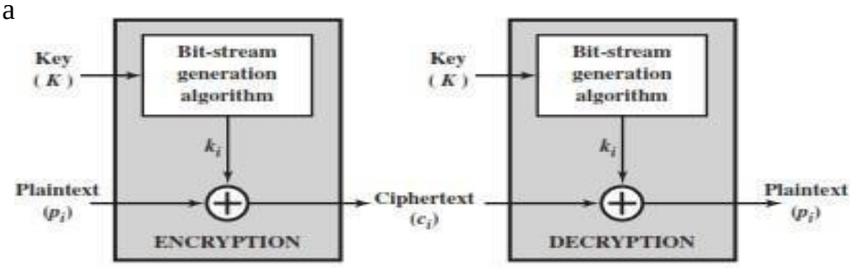
UNIT-2

Block Cipher principles:

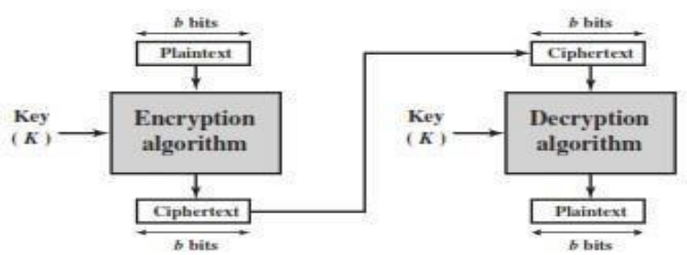
- A block cipher is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.
- Many block ciphers have a Feistel structure. Such a structure consists of a number of identical rounds of processing. In each round, a substitution is performed on one half of the data being processed, followed by a permutation that interchanges the two halves. The original key is expanded so that a different key is used for each round.
- The Data Encryption Standard (DES) has been the most widely used encryption algorithm until recently. It exhibits the classic Feistel structure. DES uses a 64-bit block and a 56-bitkey.

Stream Ciphers and Block Ciphers:

- A **stream cipher is one that encrypts a digital data stream one bit or one byte at a time. Examples of** classical stream ciphers are the auto keyed Vigenère cipher and the Vernam cipher.
- A **block cipher is** one in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length.
- Typically, a block size of 64 or 128 bits is used. As with a stream cipher, the two users share a



(a) Stream cipher using algorithmic bit-stream generator



(b) Block cipher

symmetric encryption key

The Feistel Cipher:

Feistel proposed that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers. The essence of the approach is to develop a block cipher with a key length of k bits and a block length of *n bits, allowing a total of* 2^k *possible transformations, rather than the* 2^n*! transformations available with the ideal block cipher.*

In particular, Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:

- **Substitution:** Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertextelement or group of elements.
- **Permutation:** A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.

Feistel's is a practical application of a proposal by Claude Shannon to develop a product cipher that alternates *confusion* and *diffusion* functions

FEISTEL CIPHER STRUCTURE:

The left-hand side of Figure depicts the structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key. The plaintext block is divided into two halves, L0 and R0. The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs *Li*-1 and Ri-1 derived from the previous round, as well as a subkey *Ki* derived from the overall K. In general, the subkeys *Ki are different from K and from each other.*

All rounds have the same structure. A **substitution** is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round butis parameterized by the round subkey Ki.

Permutation is performed that consists of the interchange of the two halves of the data. This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon. The exact realization of a Feistel network depends on the choice of the following parameters and design features:

- **Block size:** Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion. Traditionally, a block size of 64 bits has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.
- **Key size:** Larger key size means greater security but may decrease encryption/ decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become acommon size.
- **Number of rounds:** The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is

16rounds.

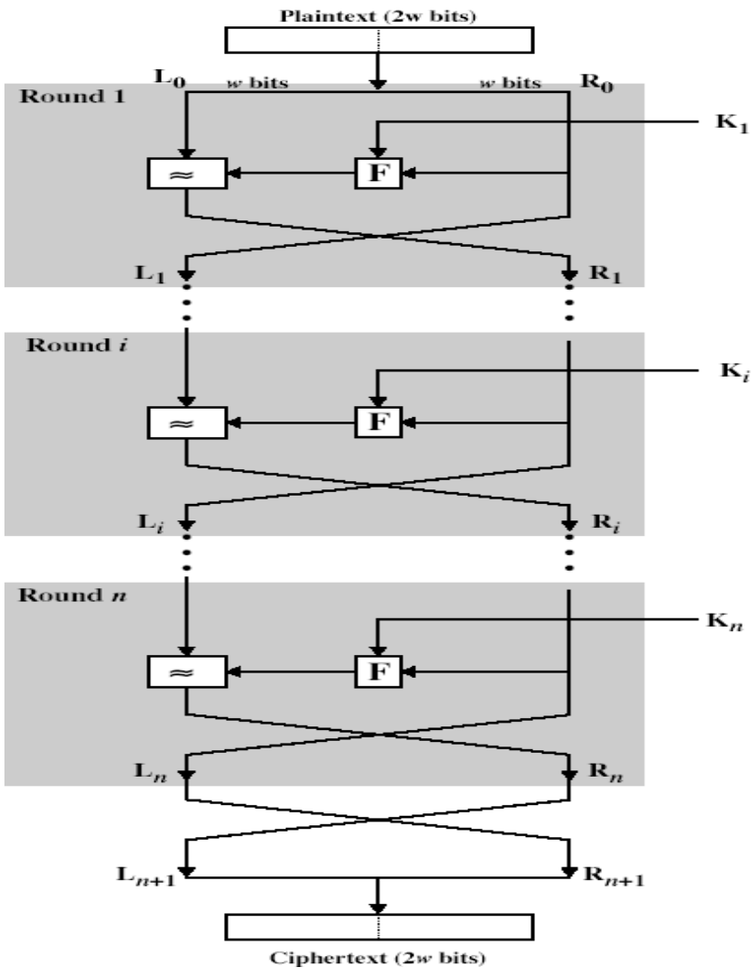


Fig: Feistel Cipher structures

- **Subkey generation algorithm**: Greater complexity in this algorithm should lead to greater difficulty ofcryptanalysis.
- **Round function F**: Again, greater complexitygenerally means greater resistance to

cryptanalysis. There are two other considerations in the design of a Feistel cipher:

- **Fast software encryption/decryption**: In many cases, encryption is embedded in applications or
  utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes aconcern.

Ease of analysis: Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, i t is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength. DES, for example, does not have an easily analysed functionality.

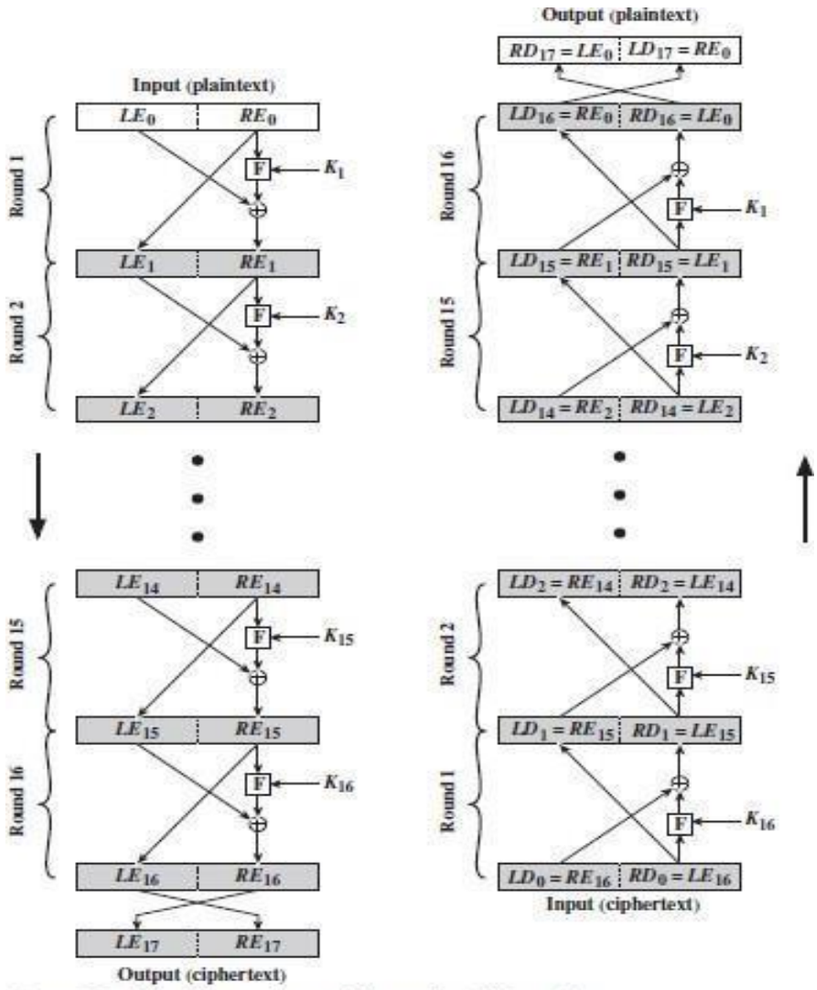


Figure 3.3 Feistel Encryption and Decryption (16 rounds)

Feistel Decryption Algorithm:

The process of decryption with a Feistel cipher is essentially the same as the encryption process. The rule is as follows:

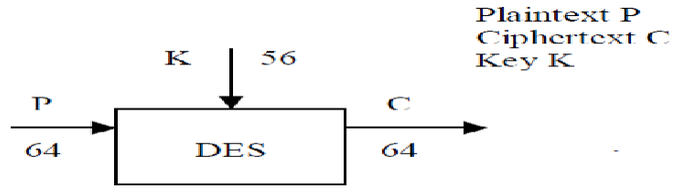
- Use the ciphertext as input to the algorithm, but use the sub keys *K* in reverse order.
- That is, use *Kn* in the first round, *Kn-1* in the second round, and so on until *K* is used in the last round. This is a nice feature because it means we need not implement two different algorithms, one for encryption and one for decryption.
- To see that the same algorithm with a reversed key order produces the correct result, which shows the encryption process going down the left-hand side and the decryption process going up the right-hand side for a 16-round algorithm.

- For clarity, we use the notation LEi and REi for data traveling through the encryption algorithm and LDi and RDi for data traveling through the decryption algorithm.
- The diagram indicates that, at every round, the intermediate value of the decryption process is equal to the corresponding value of the encryptionprocess with the two halves of the value swapped.
- After the last iteration of the encryption process, the two halves of the output are swapped, so that the ciphertext is RE16||LE16. The output of that round is the ciphertext. Now take that ciphertext and use it as input to the same algorithm. The input to the first round is RE16||LE16, which is equal to the 32-bit swap of the output of the sixteenth round of the encryption process.
- If you clearly observe that the output of the first round of the decryption process is equal to a 32-bit swap of the input to the sixteenthround of the encryptionprocess. First, consider the encryption process.

encryption process	On the decryption side,
$LE_{16} = RE_{15}$ $RE_{16} = LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})$ **in general** $LE_i = RE_{i-1}$ $RE_i = LE_{i-1} \oplus \mathrm{F}(RE_{i-1}, K_i)$	$LD_1 = RD_0 = LE_{16} = RE_{15}$ $RD_1 = LD_0 \oplus \mathrm{F}(RD_0, K_{16})$ $= RE_{16} \oplus \mathrm{F}(RE_{15}, K_{16})$ $= [LE_{15} \oplus \mathrm{F}(RE_{15}, K_{16})] \oplus \mathrm{F}(RE_{15}, K_{16})$

Data Encryption Standard:

- DES is a Symmetric-key algorithm for the encryption of electronicdata.
- DES originated at IBM in 1977& was adopted by the U.S Department of Defence. Now it is under the NIST (National Institute of Standard & Technology)
- Data Encryption Standard (DES) is a widely-used method of data encryption using a private (secret) key
- DES applies a 56-bit key to each 64-bit block of data. The process can run in several modes and involves 16rounds or operations.



Inner workings of DES:

DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher. This was a block cipher developed by the IBM cryptography researcher Horst Feistel in the early 70's. It consists of a number of rounds where each round contains bit-shuffling, non-linear substitutions (S-boxes) and exclusive OR operations. Most symmetric encryption schemes today are based on this structure (known as a Feistel network).

Overall structure

DES (and most of the other major symmetric ciphers) is based on a cipher known as the Feistel block cipher.
Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases.

- First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permutedinput*.
- This is followedby a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the **preoutput**.
- Finally, the preoutput is passed through a permutation that is the inverse of the initial permutation function, to produce the 64-bit cipher text. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher,

The right-hand portion of below shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the sixteen rounds, a *subkey* (*Ki*) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

I

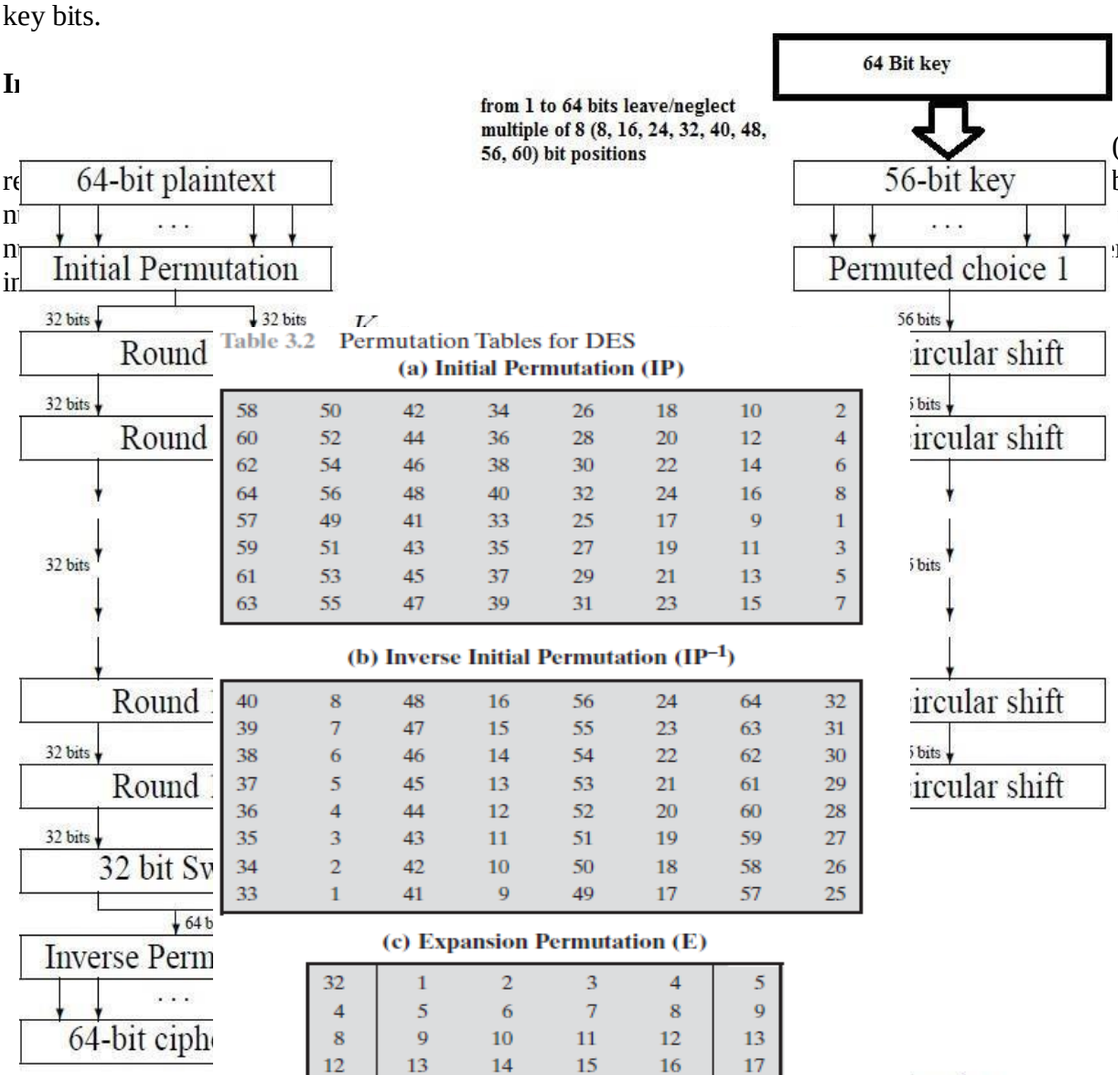


Table 3.2 Permutation Tables for DES

(a) Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(c) Expansion Permutation (E)

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

To see that these two permutation functions are indeed the inverse of each other, consider the following 64-bit input M:

M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8
M_9	M_{10}	M_{11}	M_{12}	M_{13}	M_{14}	M_{15}	M_{16}
M_{17}	M_{18}	M_{19}	M_{20}	M_{21}	M_{22}	M_{23}	M_{24}
M_{25}	M_{26}	M_{27}	M_{28}	M_{29}	M_{30}	M_{31}	M_{32}
M_{33}	M_{34}	M_{35}	M_{36}	M_{37}	M_{38}	M_{39}	M_{40}
M_{41}	M_{42}	M_{43}	M_{44}	M_{45}	M_{46}	M_{47}	M_{48}
M_{49}	M_{50}	M_{51}	M_{52}	M_{53}	M_{54}	M_{55}	M_{56}
M_{57}	M_{58}	M_{59}	M_{60}	M_{61}	M_{62}	M_{63}	M_{64}

Where stream cipher Mi is a binary digit. Then the permutation $X = (\text{IP}(M))$ is as follows:

$$\begin{array}{cccccccc}
M_{58} & M_{50} & M_{42} & M_{34} & M_{26} & M_{18} & M_{10} & M_2 \\
M_{60} & M_{52} & M_{44} & M_{36} & M_{28} & M_{20} & M_{12} & M_4 \\
M_{62} & M_{54} & M_{46} & M_{38} & M_{30} & M_{22} & M_{14} & M_6 \\
M_{64} & M_{56} & M_{48} & M_{40} & M_{32} & M_{24} & M_{16} & M_8 \\
M_{57} & M_{49} & M_{41} & M_{33} & M_{25} & M_{17} & M_9 & M_1 \\
M_{59} & M_{51} & M_{43} & M_{35} & M_{27} & M_{19} & M_{11} & M_3 \\
M_{61} & M_{53} & M_{45} & M_{37} & M_{29} & M_{21} & M_{13} & M_5 \\
M_{63} & M_{55} & M_{47} & M_{39} & M_{31} & M_{23} & M_{15} & M_7
\end{array}$$

If we then take the inverse permutation $Y = \text{IP}^{-1}(X) = \text{IP}^{-1}(\text{IP}(M))$, it can be seen that the original ordering of the bits is restored.

DETAILS OF SINGLE ROUND

Below figure shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

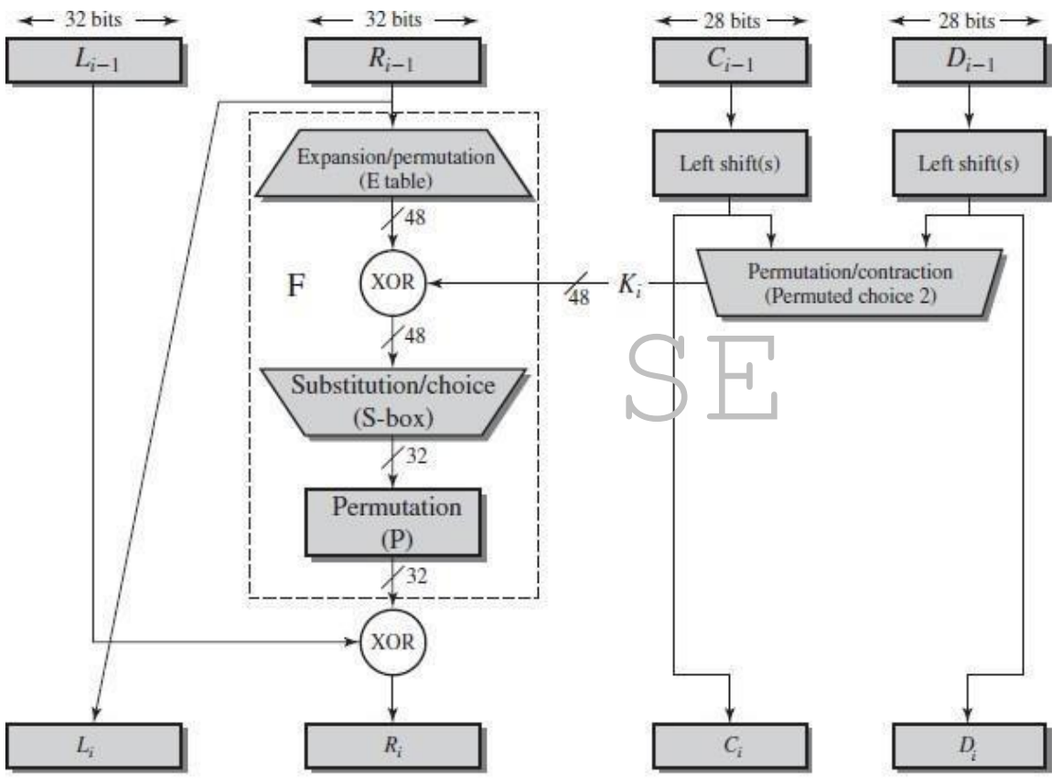


Figure :Single Round of DES Algorithm

The round key Ki is 48 bits. The *R* input is 32 bits. This *R* input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the *R* bits (Table 3.2c). The resulting 48 bits are XORed with Ki . This 48-bit result passes through substitution function that produces a 32-bit output, which is permuted as defined by Table (d). The role of the S-boxes in the function F is illustrated in Figure 3.7. The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. These transformations are defined in Table 3.3, which is interpreted as follows: The first and last bits of the input to box Si form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for. The middle four bits select one of the sixteen columns. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the

output.

For example, in S1, for input 011001, the row is 01 (row 1) and the column is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001. Each row of an S-box defines a general reversible substitution. Figure 3.2 may be useful in understanding the mapping. The figure shows the substitution for row 0 of box S1. The operation of the S-boxes is worth further comment. Ignore for the moment the contribution of the key (*Ki*). If you examine the expansion table, you see that the 32 bits of input are split into groups of 4 bits and then become groups of 6 bits by taking the outer bits from the two adjacent groups. For example, if part of the input word is
... efgh ijkl mnop ... This becomes ... defghi hijklm lmnopq ...

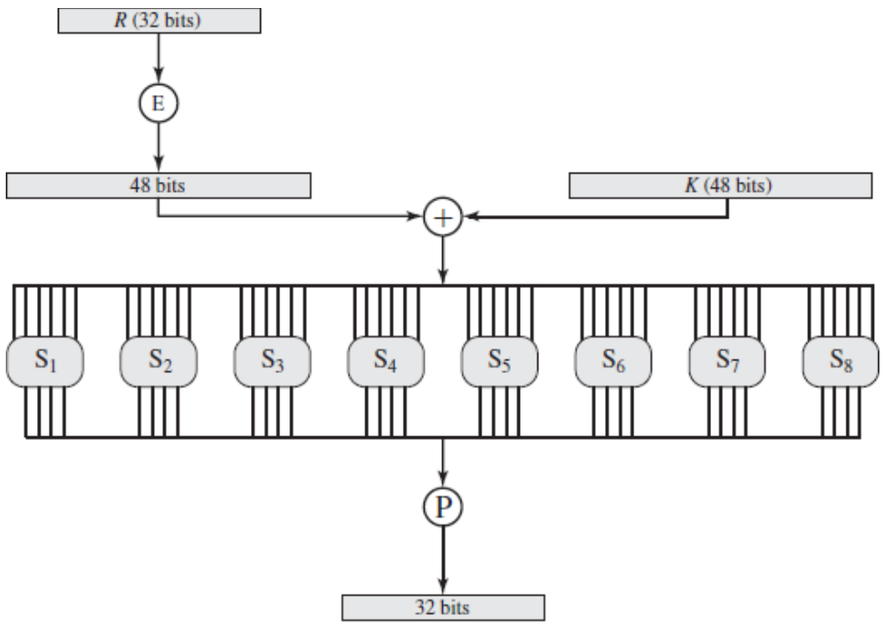


Figure 3.7 Calculation of F(R, K)

The outer two bits of each group select one of four possible substitutions (one row of an S-box). Then a 4-bit output value is substituted for the particular 4-bit input (the middle four input bits). The 32-bit output from the eight S-boxes is then permuted, so that on the next round, the output from each S-box immediately affects as many others as possible.

Substitution Boxes S: Have eight S-boxes which map 6 to 4 bits. Each S-box is actually 4 little 4 bit boxes. Outer bits 1 & 6 (**row** bits) select one rows. inner bits 2-5 (**col** bits) are substituted. Result is 8 lots of 4 bits, or 32 bits. Row selection depends on both data & key

KEY GENERATION:

Returning to above all figures, we see that a 64-bit key is used as input to the algorithm. The bits of the key are numbered from 1 through 64; every eighth bit is ignored, as indicated by the lack of shading in Table 3.4a. The key is first subjected to a permutation governed by a table labeled Permuted Choice One (Table 3.4b)
The resulting 56-bit key is then treated as two 28-bit quantities, labeled C0 and D0. At each round, Ci-1 and Di-1 are separately subjected to a circular left shift or (rotation) of 1 or 2 bits, as governed by Table 3.4d. These shifted values serve as input to the next round. They also serve as input to the part labeled Permuted Choice Two (Table 3.4c), which produces a 48-bit output that serves as input to the Function F(Ri-1, Ki).

DES:

Whatever process we following in the encryption that process is used for decryption also but the order of key is changed on input message (cipher text).
Reverse order of keys are K16, K15 ,……, K1.

The Avalanche Effect:

- A desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the ciphertext.
- In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext.

- This is referred to as the avalanche effect.

THE STRENGTH OF DES:

The Use of 56-Bit Keys:

- With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2 x 10^{16}. A brute-force attack appears impractical. Assuming that, on average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher. Diffie and Hellman postulated that the technology existed to build a parallel machine with 1 million encryption devices, each of which could perform one encryption per microsecond. This would bring the average search time down to about 10 hours.

The Nature of the DES Algorithm:

- Possibilities of cryptanalysis are done by finding the characteristics of DES algorithm.
- Learning of S-Box logic is complex.
- Weakness of the S-boxes not been discovered.

Timing Attacks:

- A timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various ciphertexts.
- A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.
- DES appears to be fairly resistant to a successful timing attack.

Block Cipher Design Principles:

There are three critical aspects of block cipher design: **the number of rounds, design of the function F, and key scheduling**.

Number of Rounds:

- The greater the number of rounds, the more difficult it is to perform cryptanalysis, even for a relatively weak F.
- In general, the criterion should be that the number of rounds is chosen so that known cryptanalytic efforts require greater effort than a simple brute-force key search attack. This criterion was certainly used in the design of DES.

Design of Function F:

- The heart of a Feistel block cipher is the function F, which provides the element of confusion in a Feistel cipher. Thus, it must be difficult to “unscramble” the substitution performed by F.
- F must be nonlinear. The more nonlinear F, the more difficult any type of cryptanalysis will be.

Key Schedule Algorithm:

- With any Feistel block cipher, the key is used to generate one subkey for each round.
- In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key.

Triple DES(3DES):

- Triple DES is simply another mode of DES operation. It takes three 64-bit keys, for an overall key length of 192 bits.
- The Triple DES then breaks the user provided key into three subkeys, padding the keys if necessary so they are each 64 bits long.
- The procedure for encryptionis exactlythe same as regular DES, but it is repeated three times. Hence the name Triple DES. The data is encrypted with the first key, decrypted with the second key, and finally encrypted again with the third key.

ADVANCED ENCRYPTION STANDARD (AES):

- The Advanced Encryption Standard (AES) was published by the National Institute of Standards and Technology (NIST) in 2001.
- AES is a block cipher intended to replace DES for commercial applications.
- It uses a 128-bit block size and a key size of 128, 192, or 256 bits.
- AES does not use a Feistel structure. Instead, each full round consists of four separate functions: byte substitution, permutation, arithmetic operations over a finite field, and XOR with a key.

Rijndael was designed to have the following characteristics:

- Resistance against all known attacks
- Speed and code compactness on a wide range of platforms
- Design simplicity

AES parameters:

Key size(words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block Size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round Key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

Inner Workings of a Round

The algorithm begins with an Add round key stage followed by 9 rounds of four stages and a tenth round of three stages. This applies for both encryption and decryption with the exception that each stage of a round the decryption algorithm is the inverse of its counterpart in the encryption algorithm. The four stages are as follows:

1. Substitute bytes
2. Shift rows
3. Mix Columns
4. Add Round Key

The tenth round simply leaves out the Mix Columns stage. The first nine rounds of the decryption algorithm consist of the following:

1. Inverse Shift rows
2. Inverse Substitute bytes
3. Inverse Add Round Key
4. Inverse Mix Columns

Again, the tenth round simply leaves out the **Inverse Mix Columns** stage. Each of these stages will now be considered in more detail.

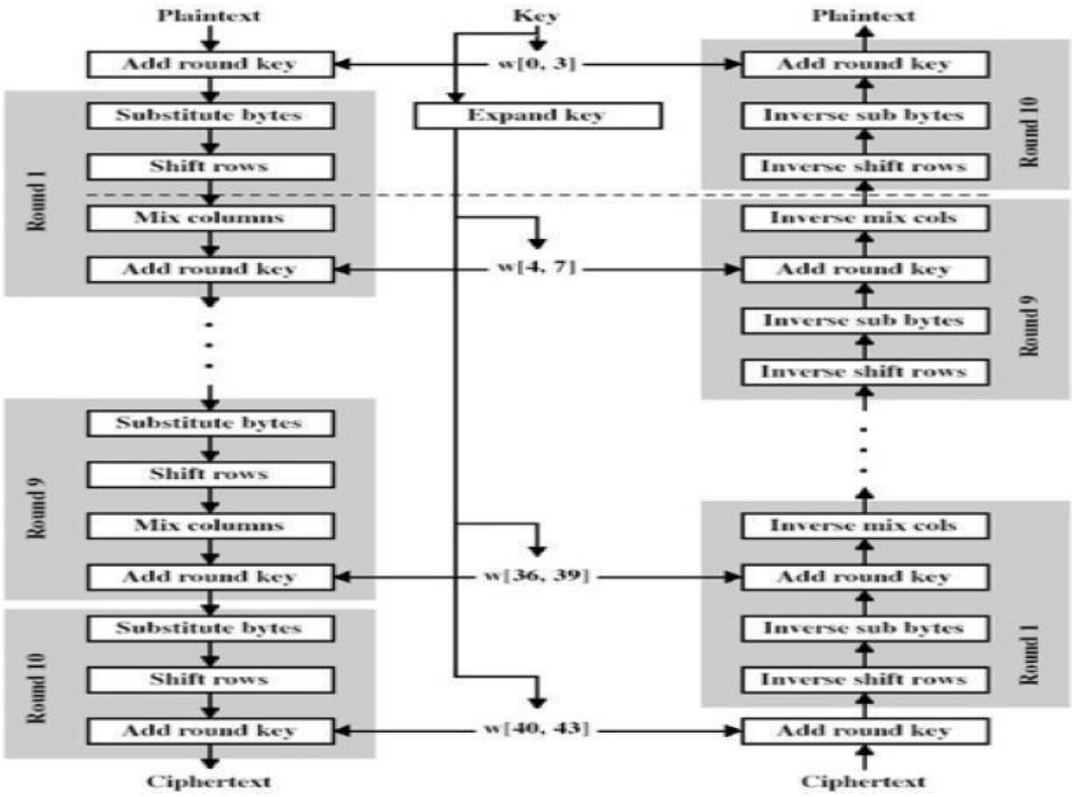


FIGURE: 7.1 overall structure of the AES algorithm

Substitute Bytes

This stage (known as SubBytes) is simply a table lookup using a 16×16 matrix of byte values called an s-box. This matrix consists of all the possible combinations of an 8-bit sequence ($2^8 = 16 \times 16 = 256$). However, the s-box is not just a random permutation of these values and there is a well- defined method for creating the s-box tables. The designers of Rijndael showed how this was done unlike the s-boxes in DES for which no rationale was given.

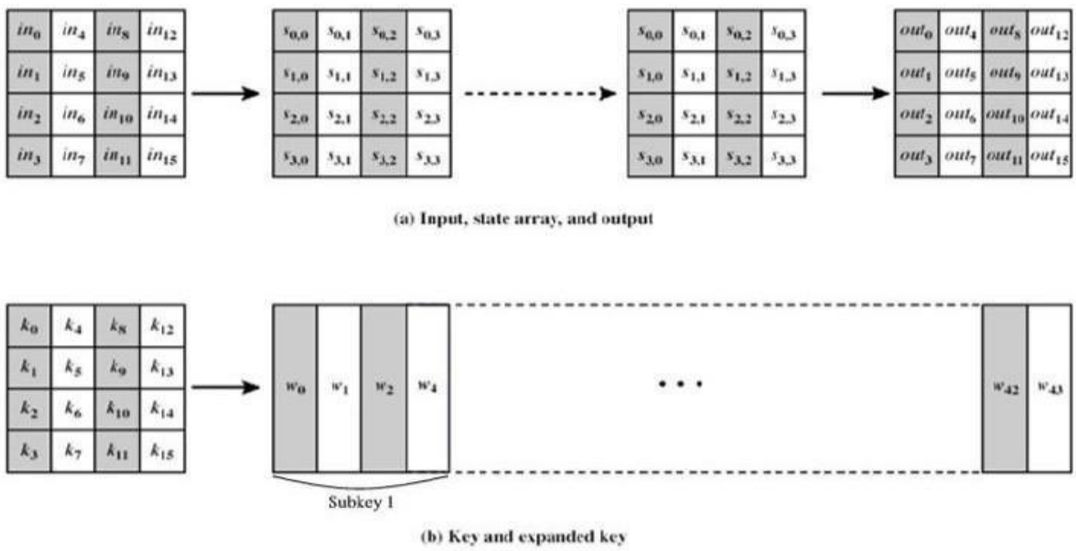


Figure 7.2: Data structures in the AES algorithm.

Again the matrix that gets operated upon throughout the encryption is known as **state**. We will be concerned with how this matrix is effected in each round. For this particular round each byte is mapped into a new byte in the following way: the leftmost nibble of the byte is used to specify a particular row of the s-box and the rightmost nibble specifies a column. For example, the byte {95} (curly brackets represent hex values in FIPS PUB 197) selects row 9 column 5 which turns out to contain the value {2A}. This is then used to update the **state** matrix. Figure 7.3 depicts this idea.

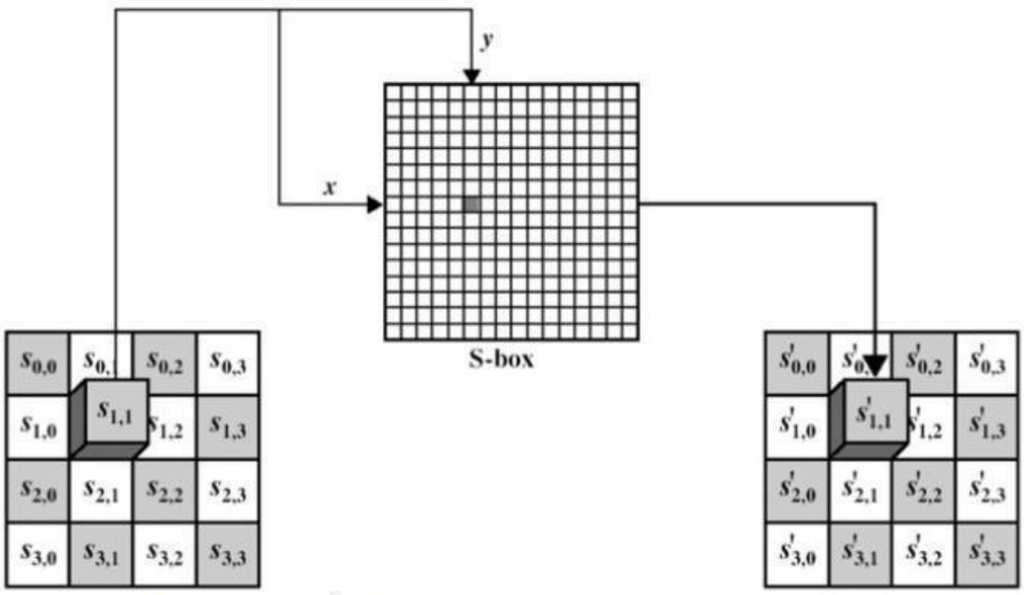


Figure 7.3: Substitute Bytes Stage of the AES algorithm.

The Inverse substitute byte transformation makes use of an inverse s-box. In this case what is desired is to select the value {2A} and get the value {95}. Table 7.4 shows the two s-boxes and it can be verified that this is in fact the case.

The s-box is designed to be resistant to known cryptanalytic attacks. Specifically, the Rijndael developers sought a design that has a low correlation between input bits and output bits, and the property that the output cannot be described as a simple mathematical function of the input. In addition, the s-box has no fixed points (s-box(a) = a) and no opposite fixed points (s-box(a) = $\bar{a}$) where is the bitwise compliment of a.

Shift Rows Transformation:

Shift row transformations are two types.

- Forward Shift row transformation which is used in encryption.
- Inverse Shift row transformation which is used in decryption.

FORWARD SHIFT ROW TRANSFORMATION:

- the first row of State matrix is not altered.
- for the second row, a 1-byte circular left shift is performed.
- for the third row, a 2-byte circular left shift is performed.
- for the fourth row, a 3-byte circular left shift is performed.

The following is an example of Shift Rows:

87	F2	4D	97
EC	6E	4C	90
4A	C3	46	E7
8C	D8	95	A6

→

87	F2	4D	97
6E	4C	90	EC
46	E7	4A	C3
A6	8C	D8	95

INVERSE SHIFT ROWS:

- Performs the circular shifts in the opposite direction for each of the last three rows, with a one- byte circular right shift for the second row and soon.

MIX COLUMNS TRANSFORMATION:

Mix columns transformations are two types.

- Forward Mix columns transformation which is used in encryption.

- Inverse Mix columns transformation which is used in decryption.

Forward Mix columns transformation:

- Forward Mix columns transformation called mix columns, operates on each column individually. Each byte of a column is mapped into a new value that is a function of all 4 bytes in that column. The transformation can be defined by the following matrix multiplication on state.

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Inverse Mix columns transformation:

- The inverse mix column transformation, called InvMixColumns, is defined by the following matrix multiplication:

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

Add Round Key Transformation:

- In the forward add round key transformation, called AddRoundKey, the 128 bits of State are bitwise XORed with the 128 bits of the round key.
- The inverse add round key transformation is identical to the forward add round key transformation, because the XOR operation is its own inverse.

47	40	A3	4C
37	D4	70	9F
94	E4	3A	42
ED	A5	A6	BC

$\oplus$

AC	19	28	57
77	FA	D1	5C
66	DC	29	00
F3	21	41	6A

=

EB	59	8B	1B
40	2E	A1	C3
F2	38	13	42
1E	84	E7	D2

AES Key Expansion:

- The 128-bit key value can be expanded into 44 words i.e. 44X32=1408bits
- In each round 4 words will be used i.e. 4x32=128 bits
- In Add round key first 4 words w0, w1, w2, w3 are used.
- In first round, w4, w5, w6, w7 are used and soon. The 128 bit key is expanded as follows
- First 128 bit key is arranged as a 4x4 matrix each value size is 8-bits
- The first 32 bits (k0, k1, k2, k3) is considered as w0.
- The first 32 bits (k4, k5, k6, k7) is considered as w1.
- The first 32 bits (k8, k9, k10, k11) is considered as w2.
- The first 32 bits (k12, k13, k14, k15) is considered as w4.
- Next 4 words w4, w5, w6, w7 are followed as

  w4=w0 $\oplus$ w3

  w5=w1 $\oplus$ w4

  w6=w2 $\oplus$w5

  w7=w3 $\oplus$w6

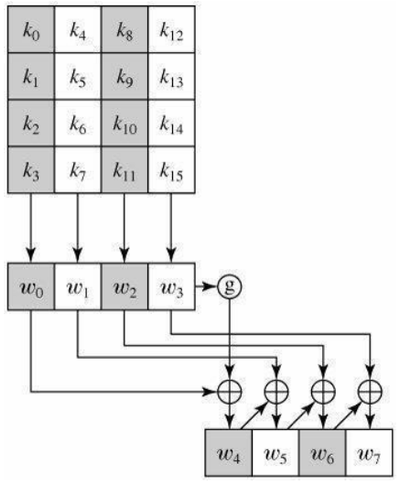


Figure. AES Key Expansion

BLOWFISH:

- Blow fish is a symmetric block cipherdeveloped by bruce schner in year 1993.
- Blow fish is designed to have following characteristics
  - ✓ Speed: Blowfish encrypts data on 32-bit microprocessor at a rate of 18 clock cycles per byte.
  - ✓ Compact: it can run in less than 5k memory.
  - ✓ Simple: very easy to implements.
  - ✓ Variably secure: the key length is variable and can be as long as 448 bits. This allows a trade-off between higher speed and higher security.
- Blowfish is a Feistel type model.

BLOWFISH ALGORITHM:

- Blowfish is Feistel type model, iterating a simple encryption function 16times.
- Blowfish block size is 64& key can be upto 448 bits.
- Blow fish encryption 64bits blocks of plaintext into 64-bit block of cipher.
- Blow fish make use of a key that ranges from 32bits to 448 bits (one to fourteen 32 bit keys).
  - The keys are stored in a k-array (one to 14 32 bits) K1,K2 Kj where $1 \leq j \leq 14$.
- That key is used to generate 18 "32 bit" subkeys & four "8*32"bits S-boxes.
  - The subkeys are stored in the p- array P1,P2, P18

There are four s-boxes (each s-box size is 8*32 bits) each with 256 32bit entries.

S1,0, S1,1;	S1,255
S2,0, S2,1;	S2,255
S3,0, S3,1;	S3,255
S4,0, S4,1;	S4,255

The steps in generating the P-array & S-boxes as follows.

Step1: Initialize first the P-array and then 4 s-boxes in order using the bits of fractional part of the constant π.

Step 2: Perform a bitwise xor of the P-array & k-array, reusing words from the k-array as needed.

Example

P1 = P1 ⊕ K1
P2 = P1 ⊕ K2
...
P14 = P14 ⊕ K14
P15 = P15 ⊕ K1

$$P16 = P1 \oplus K2$$
$$P17 = P1 \oplus K3$$
$$P18 = P1 \oplus K4$$

Step 3: Encrypt the 64 bit block of all zeros using the current P & S-arrays, Replace P1&P2 with the output of the encryption.

$$P_1, P_2 \quad = E_{P,S}[0]$$

Step 4: Encrypt the output of step 3 using the current P- and S-arrays and replace ***P3,*** and ***P4,*** with the resulting ciphertext.

$$P_3, P_4 \quad = E_{P,S}[P_1 || P_2]$$
$$\ldots$$
$$P_{17}, P_{18} \quad = E_{P,S}[P_{15} || P_{16}]$$

Step 5: Continue this process to update all elements of P and then, in order, all elements of S, using at each step the output of the continuously changing Blowfish algorithm.

$$S_{1,0}, S_{1,1} \quad = E_{P,S}[P_{17} || P_{18}]$$
$$\ldots$$
$$S_{4,254}, S_{4,255} \quad = E_{P,S}[S_{4,252} || S_{4,253}]$$

The update process can be summarized as follows

$$P_1, P_2 \quad = E_{P,S}[0]$$
$$P_3, P_4 \quad = E_{P,S}[P_1 || P_2]$$
$$\ldots$$
$$P_{17}, P_{18} \quad = E_{P,S}[P_{15} || P_{16}]$$
$$S_{1,0}, S_{1,1} \quad = E_{P,S}[P_{17} || P_{18}]$$
$$\ldots$$
$$S_{4,254}, S_{4,255} \quad = E_{P,S}[S_{4,252} || S_{4,253}]$$

Where Ep,s[Y] is the ciphertext produced by encrypting Y using Blowfish with the arrays S and P.

- A total of 521 executions of the Blowfish encryption algorithm are required to produce the final S- and P-arrays.
- Accordingly, Blowfish is not suitable for applications in which the secret key changes frequently. Further, for rapid execution, the P- and S-arrays can be stored rather than derived from the key each time the algorithm is used.
- This requires over 4 kilobytes of memory. Thus, Blowfish is not appropriate for applications with limited memory, such as smart cards.

Encryption and Decryption

Blowfish uses two primitive operations:

- Addition: Addition of words, denoted by +, is performed modulo 2^{32}.
- Bitwise exclusive-OR: This operation is denoted by $\oplus$

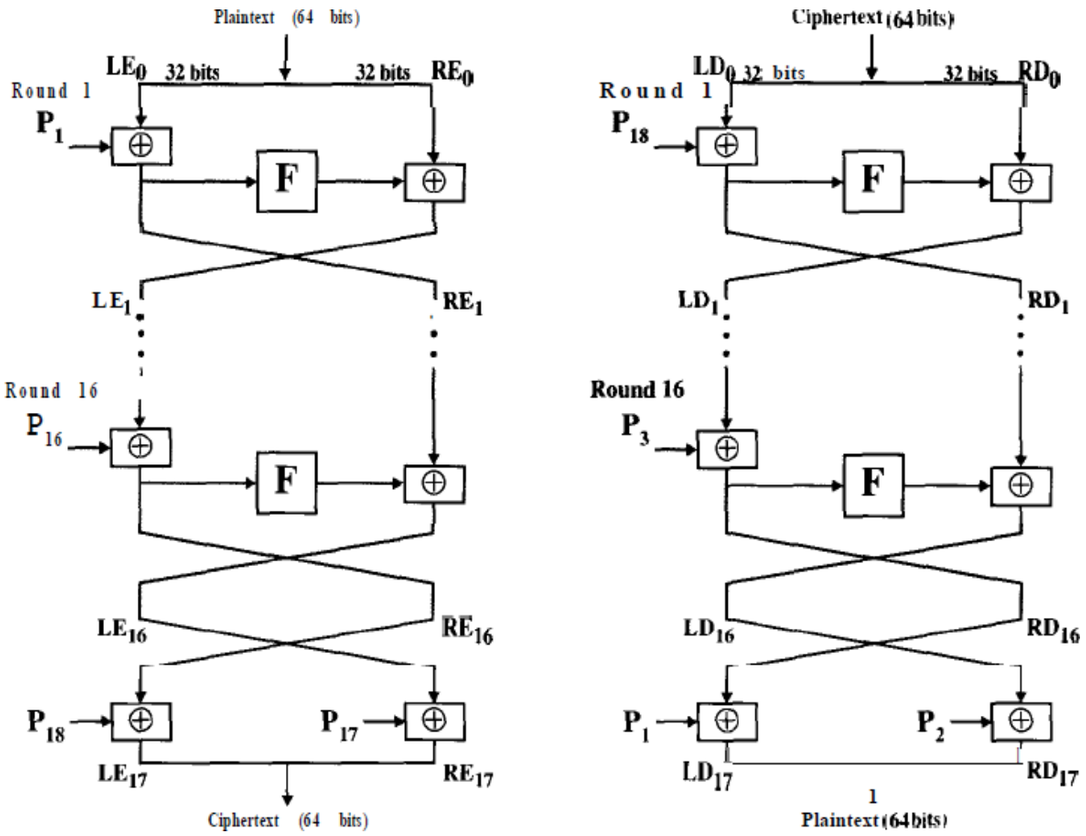


Figure Blowfish Encryption and Decryption.

In the above figure the encryption operation. The plaintext is divided into two 32-bit halves LE, and RE,. We use the variables LE, and RE, to refer to the left and right half of the data after round i has completed. The algorithm can be defined by the following pseudo code:

for $i = 1$ **to 16 do**
$RE_i = LE_{i-1} \oplus P_i$;
$LE_i = F[RE_i] \oplus RE_{i-1}$;
$LE_{17} = RE_{17} \oplus P_{18}$;
$RE_{17} = LE_{17} \oplus P_{17}$;

The function F is shown in below Figure. The 32-bit input to F is divided into 4 bytes. If we label those bytes a, b, c, and d, then the function can be defined as follows:

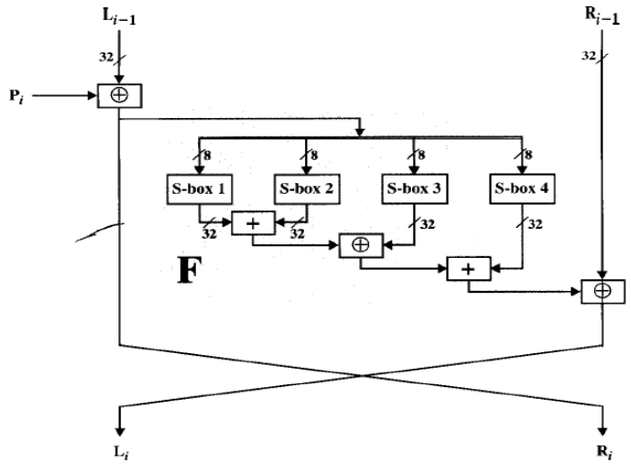


Figure Detail of Single Blowfish Round.

$$F[a, b, c, d,] = ((S_{1,a} + S_{2,b}) \oplus S_{3,c}) + S_{4,d}$$

Blowfish Decryption:

Blowfish decryption occurs in the same algorithmic direction as encryption. Rather than the reverse. The algorithm can be defined as follows:

for $i = 1$ to 16 do
$RD_i = LD_{i-1} \oplus P_{19-i}$;
$LD_i = F[RD_i] \oplus RD_{i-1}$;
$LD_{17} = RD_{17} \oplus P_1$;
$RD_{17} = LD_{17} \oplus P_2$;

Advantages or features of blowfish:

- A brute-force attack is even more difficult than may be apparent from the key length because of the time-consuming subkey-generation process. A total of 522 executions of the encryption algorithm are required to test a single key.
- The function F gives Blowfish the best possible avalanche affect for a Feistel network: In round i, every bit of Li-1, affects every bit of Ri-1. In addition, every subkey bit is affected by every key bit. and therefore F has a perfect avalanche effect between the key (P,) and the right half of the data (R,) after every round.
- Every bit of the input to F is only used as input to one S-box. In contrast. In DES, many bits are used as inputs to two S-boxes. which strengthens the algorithm considerably against differential attacks. Schneier felt that this added complexity was not necessary with key-dependent S-boxes.
- Unlike in CAST, the function F in Blowfish is not round dependent. Schneier felt that such dependency did not add any cryptographic merit, given that the P-array substitution is already round dependent.

Block Cipher Modes of Operation:

A block cipher algorithm is a basic building block for providing data security. To apply a block cipher in a variety of applications, different "modes of operation" have been defined by NIST. In essence, a mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream. The modes are intended to cover virtually all the possible applications of encryption for which a block cipher could be used.

Electronic Codebook Mode:

The simplest mode is the electronic codebook (ECB) mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key (Figure a & b). The term codebook is used because, for a given key, there is a unique ciphertext for every b-bit block of plaintext. For a message longer than b bits, the procedure is simply to break the message into b-bit blocks, padding the last block if necessary. Decryption is performed one block at a time, always using the same key. In Figure, the plaintext (padded as necessary) consists of a sequence of b-bit blocks, P1, P2... PN; the corresponding sequence of ciphertext blocks is C1, C2,..., CN.

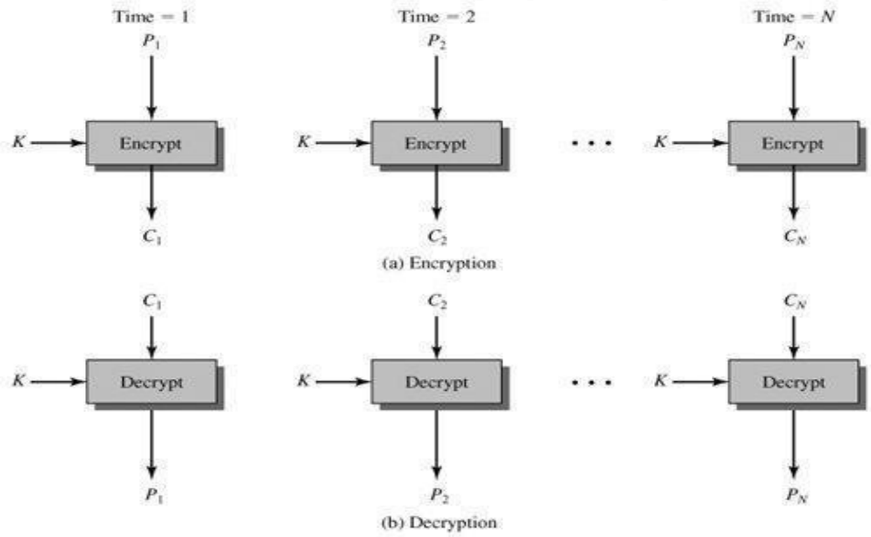


Figure. Electronic Codebook (ECB) Mode

The ECB method is ideal for a short amount of data, such as an encryption key. Thus, if you want to transmit a DES key securely, ECB is the appropriate mode to use. The most significant characteristic of ECB is that the same b-bit block of plaintext, if it appears more than once in the message, always produces the same ciphertext.

For lengthy messages, the ECB mode may not be secure. If the message is highly structured, it may be possible for a cryptanalyst to exploit these regularities. For example, if it is known that the message always starts out with certain predefined fields, then the cryptanalyst may have a number of known plaintext-ciphertext pairs to work with. If the message has repetitive elements, with a period of repetition a multiple of b bits, then these elements can be identified by the analyst. This may help in the analysis or may provide an opportunity for substituting or rearranging blocks.

Cipher Block Chaining Mode:

To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block, if repeated, produces different ciphertext blocks. A simple way to satisfy this requirement is the cipher block chaining (CBC) mode.

In this scheme, the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block; the same key is used for each block. In effect, we have chained together the processing of the sequence of plaintext blocks. The input to the encryption function for each plaintext block bears no fixed relationship to the plaintext block. Therefore, repeating patterns of *b* bits are not exposed.

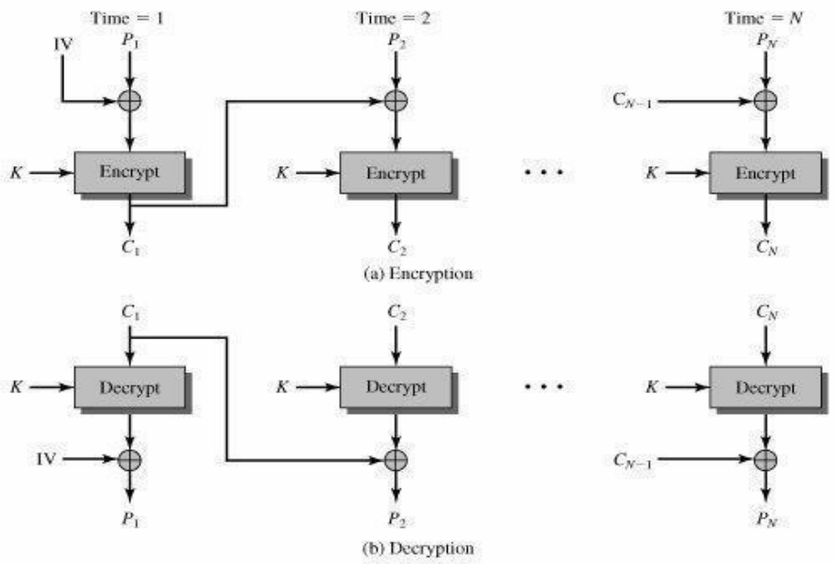


Figure: Cipher Block Chaining (CBC) Mode

For decryption, each cipher block is passed through the decryption algorithm. The result is XORed with the preceding ciphertext block to produce the plaintext block. To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext. On decryption, the IV is XORed with the output of the decryption algorithm to recover the first block of plaintext. The IV is a data block that is that same size as the cipher block. The IV must be known to both the sender and receiver but be unpredictable by a third party. For maximum security, the IV should be protected against unauthorized changes. This could be done by sending the IV using ECB encryption. Because of the chaining mechanism of CBC, it is an appropriate mode for encrypting messages of length greater than *b bits.* CBC mode can be used for authentication.

Cipher Feedback Mode:

The DES scheme is essentially a block cipher technique that uses b-bit blocks. However, it is possible to convert DES into a stream cipher, using either the cipher feedback (CFB) or the output feedback mode. Figure depicts the CFB scheme. In the figure, it is assumed that the unit of transmission is s bits; a common value is s = 8. As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext. In this case, rather than units of b bits, the plaintext is divided into segments of s bits.

First, consider **encryption**. The input to the encryption function is a b-bit shift register that is initially set to some initialization vector (IV). The leftmost (most significant) s bits of the output of the encryption function are XORed with the first segment of plaintext P1 to produce the first

unit of ciphertext C, which is then transmitted. In addition, the contents of the shift register are shifted left by s bits and C is placed in the rightmost (least significant) s bits of the shift register. This process continues until all plaintext units have been encrypted.

For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit.

Let Ss(X) be defined as the most significant s bits

of X. Then C1 = P1⊕ Ss[E(K,IV)]

Therefore,

$$P1 = C1 \oplus Ss\,[E(K, IV)]$$

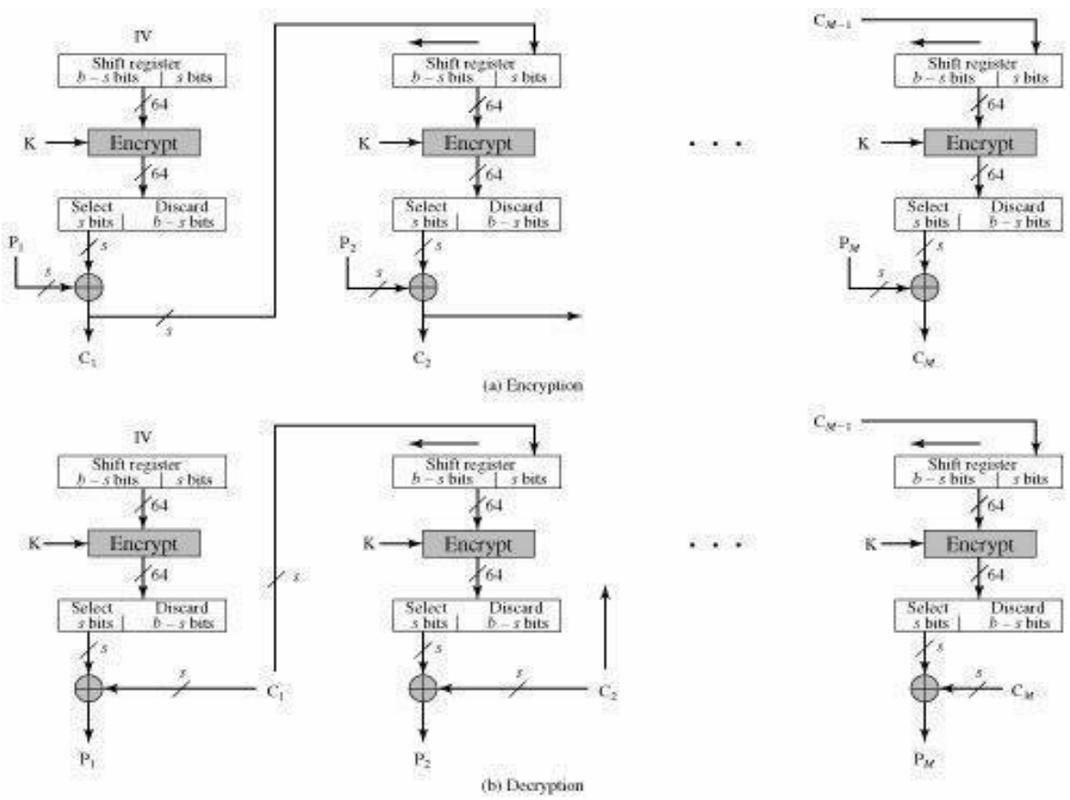


Figure: s-bit Cipher Feedback (CFB) Mode

Output Feedback Mode:

The output feedback (OFB) mode is similar in structure to that of CFB, as illustrated in Figure. As can be seen, it is the output of the encryption function that is fed back to the shift register in OFB, whereas in CFB the ciphertext unit is fed back to the shift register. One advantage of the OFB method is that bit errors in transmission do not propagate. For example, if a bit error occurs in C1 only the recovered value of is P1 affected; subsequent plaintext units are not corrupted. With CFB, C1 also serves as input to the shift register and therefore causes additional corruption downstream. The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than is CFB.

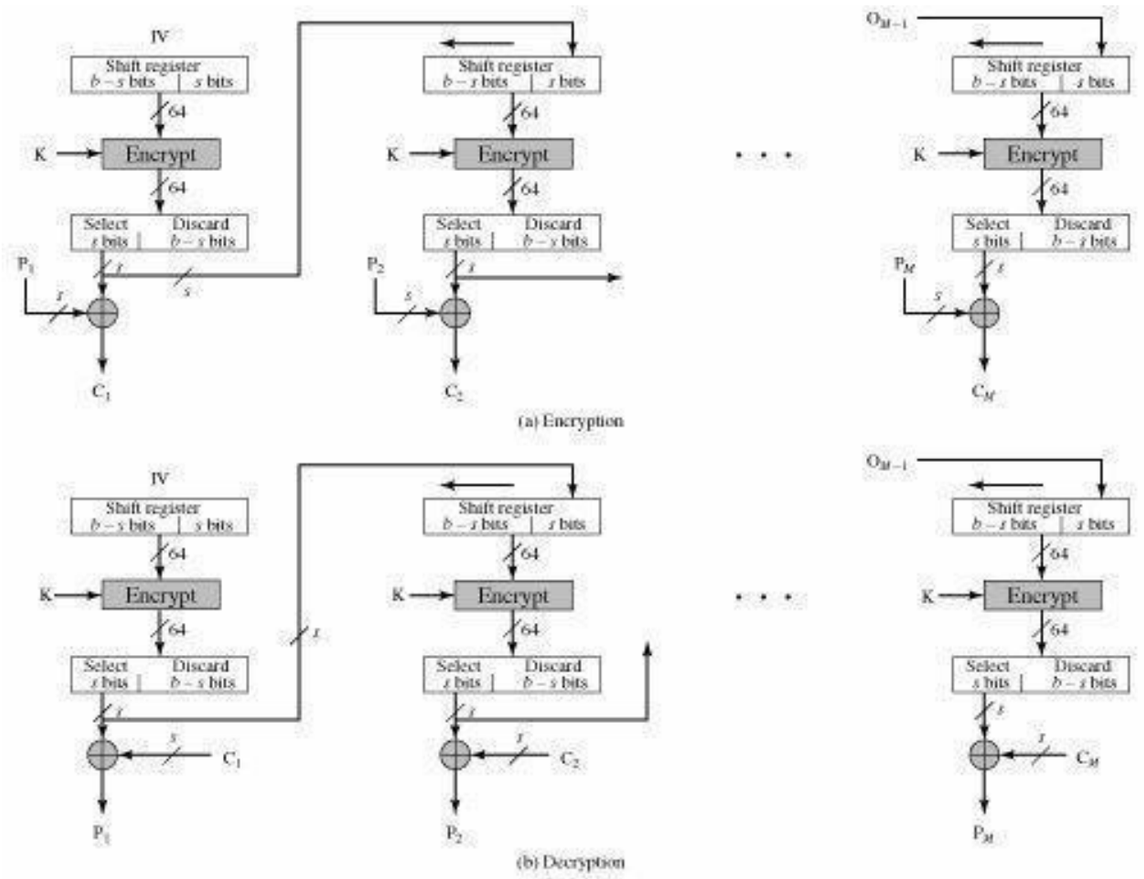


Figure: S-bit Output Feedback (OFB) Mode

Counter Mode:

In CTR mode a counter, equal to the plaintext block size is used. The only requirement is that the counter value must be different for each plaintext block that is encrypted. Typically, the counter is initialized to some value and then incremented by 1 for each subsequent block. For encryption, the counter is encrypted and then XOR end with the plaintext block to produce the ciphertext block; there is no chaining. For decryption, the same sequence of counter values is used, with each encrypted counter XOR end with a ciphertext block to recover the corresponding plaintext block.

Advantages:

1. Hardware efficiency
2. Software efficiency
3. Preprocessing
4. Random access
5. Provable security
6. Simplicity

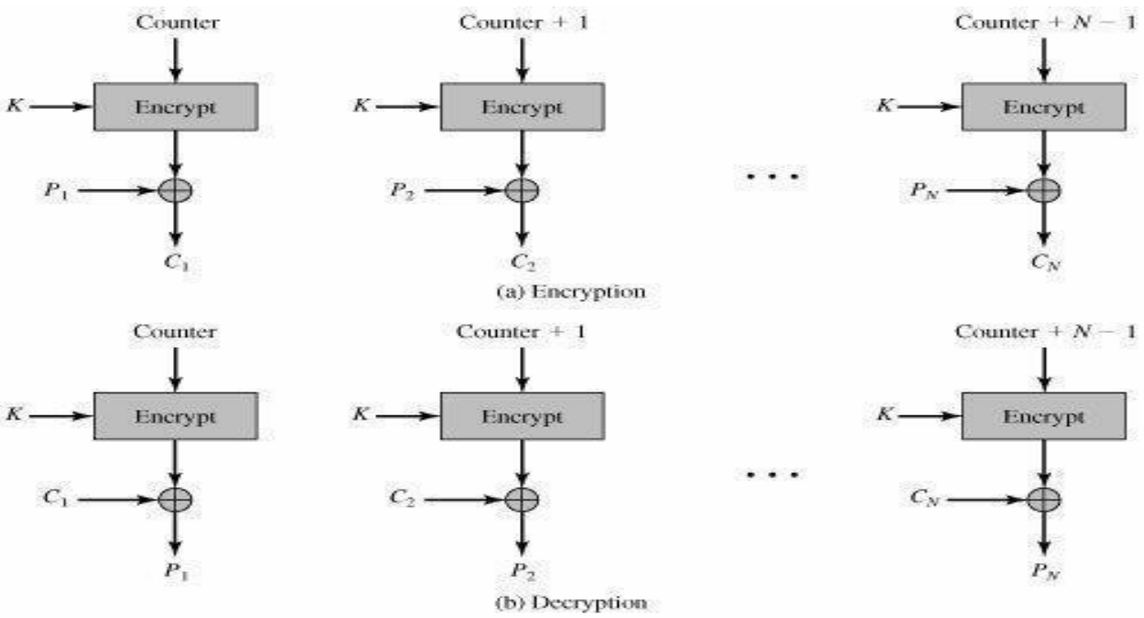


Figure: Counter (CTR) Mode

STREAM CIPHER:

A typical stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time.

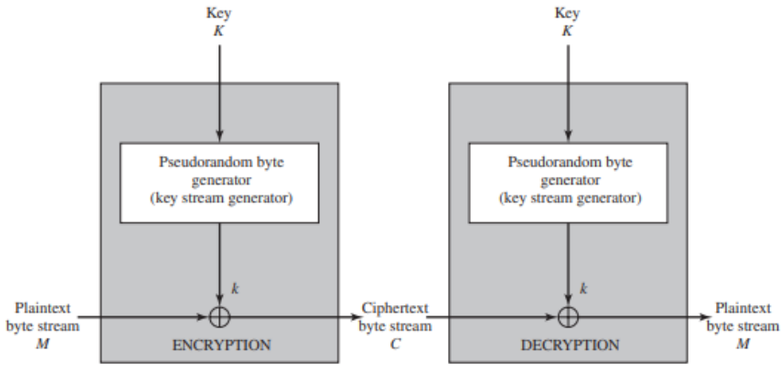


Figure 7.7 Stream Cipher Diagram

Figure: representative diagram of stream cipher structure.

In this structure, a key is input to a pseudorandom bit generator that produces a stream of 8-bit numbers that are apparently random. The output of the generator, called a keystream, is combined one byte at a time with the plaintext stream using the bitwise exclusive-OR (XOR) operation.

Example, if the next byte generated by the generator is 01101100 and the next plaintext byte is 11001100, then the resulting ciphertext byte is

```
  11001100 plaintext
⊕ 01101100 key stream
  10100000 ciphertext
```

Decryption requires the use of the same pseudorandom sequence:

```
  10100000 ciphertext
⊕ 01101100 key stream
  11001100 plaintext
```

Important design considerations for a stream cipher:

1. The encryption sequence should have a large period. A pseudorandom number generator uses a function that produces a deterministic stream of bits that eventually repeats. The longer the period of repeat the more difficult it will be to do cryptanalysis. This is essentially the same consideration that was discussed with reference to the Vigenère cipher, namely that the longer the keyword the more difficult the cryptanalysis.
2. The keystream should approximate the properties of a true random number stream as close as possible. For example, there should be an approximately equal number of 1s and 0s. If the

keystream is treated as a stream of bytes, then all of the 256 possible byte values should appear approximately equally often. The more random-appearing the keystream is, the more randomized the ciphertext is, making cryptanalysis more difficult.

3. Note from Figure 7.7 that the output of the pseudorandom number generator is conditioned on the value of the input key. To guard against brute-force attacks, the key needs to be sufficiently long. The same considerations that apply to block ciphers are valid here. Thus, with current technology, a key length of at least 128 bits is desirable.

With a properly designed pseudorandom number generator, a stream cipher can be as secure as a block cipher of comparable key length. A potential **advantage** of a stream cipher is that stream ciphers that do not use block ciphers as a building block are typically faster and use far less code than do block ciphers.

RC4:

RC4 is a stream cipher designed in 1987 by Ron Rivest for RSA Security. It is a variable key size stream cipher with byte-oriented operations. The algorithm is based on the use of a random permutation. The RC4 algorithm is remarkably simple and quite easy to explain.

A variablelength key of from 1 to 256 bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S, with elements S[0],S[1], c,S[255]. At all times, S contains a permutation of all 8-bit numbers from 0 through 255. For encryption and decryption, a byte k is generated from S by selecting one of the 255 entries in a systematic fashion. As each value of k is generated, the entries in S are once again permuted.

Initialization of S

To begin, the entries of S are set equal to the values from 0 through 255 in ascending order; that is, S[0] = 0, S[1] = 1, c, S[255] = 255 . A temporary vector, T, is also created. If the length of the key K is 256 bytes, then K is transferred to T. Otherwise, for a key of length keylen bytes, the first keylen elements of T are copied from K, and then K is repeated as many times as necessary to fill out T. These preliminary operations can be summarized as

```
/* Initialization */
for i = 0 to 255 do
S[i] = i;
T[i] = K[i mod keylen];
```

Next we use T to produce the initial permutation of S. This involves starting with S[0] and going through to S[255], and for each S[i], swapping S[i] with another byte in S according to a scheme dictated by T[i]:

```
/* Initial Permutation of S */
j = 0;
for i = 0 to 255 do
 j = (j + S[i] + T[i]) mod 256;
Swap (S[i], S[j]);
```

Because the only operation on S is a swap, the only effect is a permutation. S still contains all the numbers from 0 through 255.

Once the S vector is initialized, the input key is no longer used. Stream generation involves cycling through all the elements of S[i], and for each S[i], swapping S[i] with another byte in S according to a scheme dictated by the current configuration of S. After S[255] is reached, the process continues, starting over again at S[0]:

```
/* Stream Generation */
i, j = 0;
while (true)
 i = (i + 1) mod 256;
 j = (j + S[i]) mod 256;
Swap (S[i], S[j]);
t = (S[i] + S[j]) mod 256;
```

k = S[t];

To encrypt, XOR the value k with the next byte of plaintext. To decrypt, XOR the value k with the next byte of ciphertext. Following figure illustrates the RC4 logic.

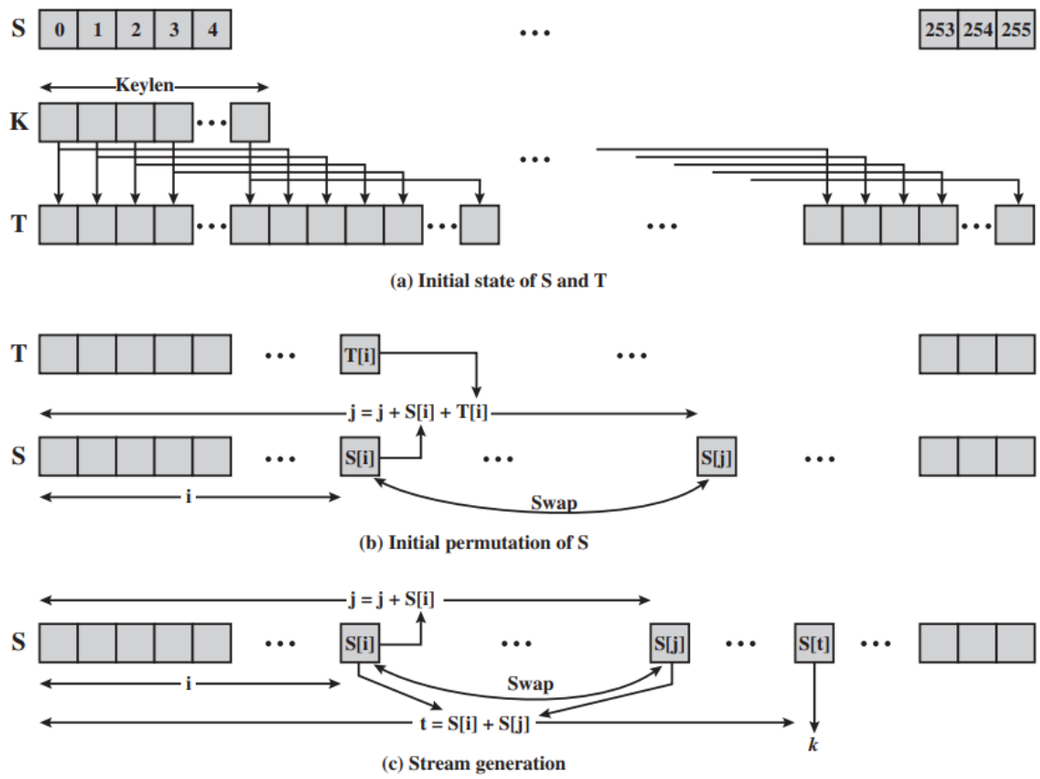


Figure 7.8 RC4

PUBLIC KEY CRYPTOGRAPHY:

Introduction:

- Asymmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the different keys - one a public key and one a private key. It is also known as public-key encryption.
- Asymmetric encryption transforms plaintext into ciphertext using a one of two keys and an encryption algorithm. Using the paired key and a decryption algorithm, the plaintext is recovered from the ciphertext.
- Asymmetric encryption can be used for confidentiality, authentication, or both.
- The most widely used public-key cryptosystem is RSA.

Principles of Public-Key Cryptosystems:

The concept of public key cryptography in invented for two most difficult problems of Symmetric key encryption.

- **key distribution** – how to have secure communications in general without having to trust a KDC (key distribution center) with yourkey.
- **digital signatures** – how to verify a message comes intact from the claimed sender.

Public-Key Cryptosystems:

A public-key encryption scheme has six ingredients

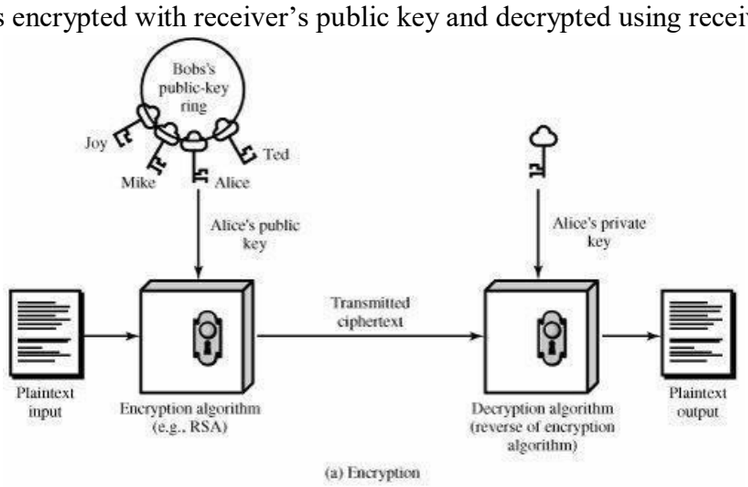
- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

The essential steps are the following:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. Each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

ENCRYPTION:

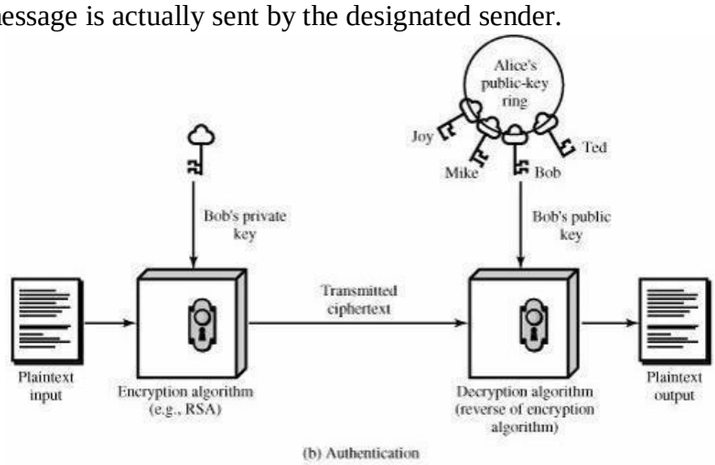
The plaintext is encrypted with receiver's public key and decrypted using receiver private key.



(a) Encryption

AUTHENTICATION:

- Plaintext is encrypted is sender's private key and decrypted using sender's publickey.
- The act of messages ciphertext getting decrypted by sender's public key is the proof that the message is actually sent by the designated sender.



(b) Authentication

Difference between symmetric and public key encryption:

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

- Examples for conventional encryption are DES, AES,IDEA and Blowfish.
- Examples for public key encryption are RSA, Diffie-Hellman, Elliptic Curve Cryptography.

There is some source A that produces a message in plaintext, X =[X1,X2,..., XM,]. The M elements of X are letters in some finite alphabet. The message is intended for destination B.

B generates a related pair of keys: a public key, PUb, and a private key, PRb. PRb is known only to B, whereas PUb is publicly available and therefore accessible by A.

With the message X and the encryption key PUb as input, A forms the ciphertext

Y = [Y1, Y2,...,YN]:
Y = E(PUb, X)

The intended receiver, in possession of the matching private key, is able to invert the transformation:
X = D(PRb, Y)

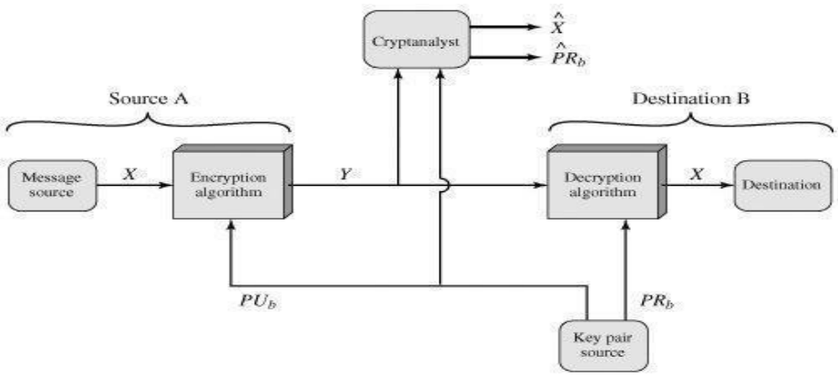


Figure: public key cryptosystems: Secrecy (or) confidentiality

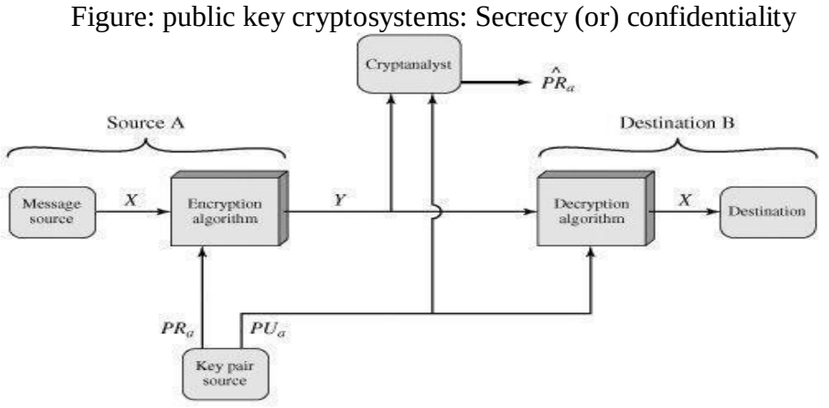


Figure: Public-Key Cryptosystem: Authentication

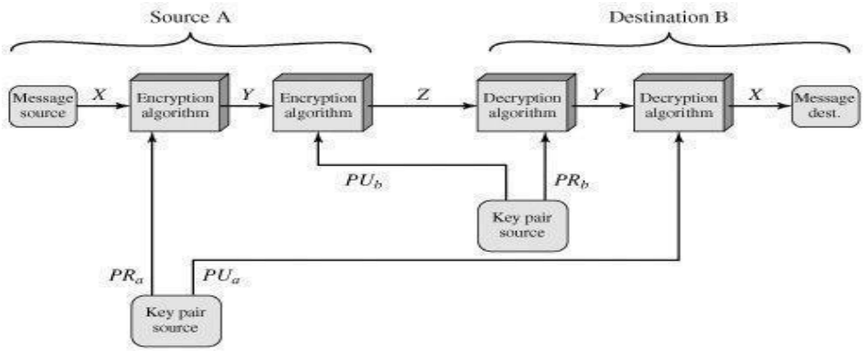


Figure: Public-Key Cryptosystem: Authentication and Secrecy

Applications for Public-Key Cryptosystems:

- **Encryption/decryption:** The sender encrypts a message with the recipient's public key.
- Digital signature: The sender "signs"a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message.
- **Key exchange:** Two sides cooperate to exchange a session key.

Algorithm	Encryption/Decryption	Digital Signature	Key Exchange
RSA	Yes	Yes	Yes
Elliptic Curve	Yes	Yes	Yes
Diffie-Hellman	No	No	Yes
DSS	No	Yes	No

Requirements for Public-Key Cryptography:

1. It is computationally easy for a party B to generate a pair (public key PUb, private key PRb).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext: C = E(PUb, M)
3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message: M= D(PRb, C) = D[PRb, E(PUb, M)]
4. It is computationally infeasible for an adversary, knowing the public key, PUb, to determine the private key,PRb.
5. It is computationally infeasible for an adversary, knowing the public key, PUb, and a ciphertext, C, to recover the original message, M.

We can add a sixth requirement that, although useful, is not necessary for all public-key applications: The two keys can be applied in either order:
M = D[PUb, E(PRb, M)] = D[PRb, E(PUb, M)]

RSA Algorithm

- It is the most common public key algorithm.
- This RSA name is get from its inventors first letter (Rivest (R), Shamir (S) and Adleman (A)) in the year 1977.
- The RSA scheme is a block cipher in which the plaintext & ciphertext are integers between 0 and n-1 for some **n**.
- A typical size for **n** is 1024 bits or 309 decimal digits. That is, n is less than 2^{1024}

Description of the Algorithm:

- RSA algorithm uses an expression with exponentials.

- In RSA plaintext is encrypted in blocks, with each block having a binary value less than some number n. that is, the block size must be less than or equal to **log2(n)**
- RSA uses two exponents e and d where e public and d private.
- Encryption and decryption are of following form, for some PlainText M and CipherText block C

$$C = M^e \bmod n$$

$$M = C^d \bmod n$$

M=Cd mod = (Me mod n) d mod n =(Me)d mod n= Med mod n

Both sender and receiver must know the value of n.

The sender knows the value of **e** & only the receiver knows the value of **d** thus this is a public key encryption algorithm with a

Publc key PU={e, n} Private key PR={d, n}

Steps of RSA algorithm:

Step 1: Select 2 prime numbers p & q
Step 2: Calculate n=pq
Step 3: Calculate Ø(n)=(p-1)(q-1)
Step 4: Select or find integer e (public key) which is relatively prime to Ø(n). ie., e with gcd (Ø(n), e)=1 where 1<e< Ø(n).
Step 5: Calculate "d" (private key) by using following condition. d< Ø(n)
Step 6: Perform encryption by using $ed \equiv 1 \bmod \text{Ø(n)}$
Step 7: perform Decryption by using $M = C^d \bmod n$

Example:

1. Select two prime numbers, $p = 17$ and $q = 11$.
2. Calculate $n = pq = 17 \times 11 = 187$.
3. Calculate $\text{Ø}(n) = (p - 1)(q - 1) = 16 \times 10 = 160$.
4. Select e such that e is relatively prime to $\text{Ø}(n) = 160$ and less than $\text{Ø}(n)$; we choose $e = 7$.
5. Determine d such that $de \equiv 1 \pmod{160}$ and $d < 160$.The correct value is $d = 23$, because $23 * 7 = 161 = (1 \times 160) + 1$;

d can be calculated using the extended Euclid"s algorithm

6. The resulting keys are public key $PU = \{7, 187\}$ and private key $PR = \{23, 187\}$.

The example shows the use of these keys for a plaintext input of $M = 88$. For encryption, we need to calculate $C = 88^7 \bmod 187$. Exploiting the properties of modular arithmetic, we can do this as follows.

$$88^7 \bmod 187 = [(88^4 \bmod 187) \times (88^2 \bmod 187) \times (88^1 \bmod 187)] \bmod 187$$

$$88^1 \bmod 187 = 88$$

$$88^2 \bmod 187 = 7744 \bmod 187 = 77$$

$$88^4 \bmod 187 = 59{,}969{,}536 \bmod 187 = 132$$

$$88^7 \bmod 187 = (88 \times 77 \times 132) \bmod 187 = 894{,}432 \bmod 187 = 11$$

For decryption, we calculate $M = 11^{23} \bmod 187$:

$$11^{23} \bmod 187 = [(11^1 \bmod 187) \times (11^2 \bmod 187) \times (11^4 \bmod 187) \times (11^8 \bmod 187) \times (11^8 \bmod 187)] \bmod 187$$

$$11^1 \bmod 187 = 11$$

$$11^2 \bmod 187 = 121$$

$$11^4 \bmod 187 = 14{,}641 \bmod 187 = 55$$

$$11^8 \bmod 187 = 214{,}358{,}881 \bmod 187 = 33$$

$$11^{23} \bmod 187 = (11 \times 121 \times 55 \times 33 \times 33) \bmod 187 = 79{,}720{,}245 \bmod 187 = 88$$

The Security of RSA

Four possible approaches to attacking the RSA algorithm are

• **Brute force:** This involves trying all possible private keys.
• **Mathematical attacks:** There are several approaches, all equivalent in effort to factoring the product of two primes.
• **Timing attacks:** These depend on the running time of the decryption algorithm.
• **Chosen ciphertext attacks:** This type of attack exploits properties of the RSA algorithm.

Trapdoor one-way function

- A trapdoor function is a function that is easy to perform one way, but has a secret that is required to perform the inverse calculation efficiently.
- That is, if f is a trapdoor function, then y=f(x) is easy to compute, but x=f−1(y) is hard to compute without some special knowledge k. Given k, then it is easy to computey=f−1(x,k).
- The analogy to a "trapdoor" is something like this: It's easy to fall through a trapdoor, but it's very hard to climb back out and get to where you started unless you have a ladder.
- An example of such trapdoor one-way functions may be finding the prime factors of large numbers. Nowadays, this task is practically infeasible.
- On the other hand, knowing one of the factors, it is easy to compute the other ones.
- For example: RSA is a one-way trapdoor function

Diffie-Hellman Key Exchange:

- Diffie-Hellman key exchange is the first published public key algorithm
- This Diffie-Hellman key exchange protocol is also known as exponential key agreement. And it is based on mathematical principles.
- The purpose of the algorithm is to enable two users to exchange a key securely that can then be used for subsequent encryption of messages.
- This algorithm itself is limited to exchange of the keys.
- This algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.
- The discrete logarithms are defined in this algorithm in the way of define a primitive root of a prime number.
- **Primitive root:** we define a primitive root of a prime number P as one whose power generate all the integers from 1 to P-1 that is if **'a'** is a primitive root of the prime number P, then the numbers are distinct and consist of the integers form 1 through P-1 in some permutation.

$$a \bmod P,\ a^2 \bmod P,\ a^3 \bmod P,\ \ldots..\ a^{P-1} \bmod P$$

For any integer **b** and **a**, here **a** is a primitive root of prime number P, then

$$b \equiv a^i \bmod P \qquad 0 \le i \le (P-1)$$

The exponent i is refer as discrete logarithm or index of b for the base a, mod P. The value denoted as **ind a,p(b)**

Algorithm for Diffie-Hellman Key Exchange:

Step 1: Select global public numbers q, α
q- Prime number
α- primitive root of q and α< q.

Step 2: if A & B users wish to exchange a key

a) User A select a random integer XA<q and computes
b) User B independently select a random integer XB <q and computes
c) Each side keeps the X value private and Makes the Y value available publicly to the outer side.

$$\boxed{K = (Y_B)^{X_A} \bmod q} \qquad Y_A = \alpha^{X_A} \bmod q$$

$$Y_B = \alpha^{X_B} \bmod q$$

Step3: User A Computes the key as

User B Computes the key as $K = (Y_A)^{X_B} \bmod q$

Step 4: two calculation produce identical results
The result is that the two sides have exchanged a secret key.

Select Global public Elements
q →Prime number
α →primitive Root of q

Here is an example. Key exchange is based on the use of the prime number $q = 353$ and a primitive root of 353, in this case $\alpha = 3$. A and B select secret keys $X_A = 97$ and $X_B = 233$, respectively. Each computes its public key:

A computes $Y_A = 3^{97} \bmod 353 = 40$.
B computes $Y_B = 3^{233} \bmod 353 = 248$.

After they exchange public keys, each can compute the common secret key:

A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.
B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

We assume an attacker would have available the following information:

$$q = 353; \alpha = 3; Y_A = 40; Y_B = 248$$

Example:

MAN-in the Middle Attack (MITM)

Definition:

A man in the middle attack is a form of eavesdropping where communication between two users is monitored and modified by an unauthorized party. Generally the attacker actively eavesdrops by intercepting (stopping) a public key message exchange. The Diffie- Hellman key exchange is insecure against a "Man in the middle attack".

Suppose user A & B wish to exchange keys, and D is the adversary (opponent). The attack proceeds as follows.

1. D prepares for the attack by generating two random private keys XD1 & XD2 and then computing the corresponding public keys YD1 and YD2.
2. A transmits YA to B
3. D intercepts YA and transmits YD1 to B. and D also calculates $K2 = (Y_A)^{X_{D2}} \bmod q$.
4. B receives YD1 & calculate $K1 = (Y_{D1})^{X_B} \bmod q$.
5. B transmits YB to A
6. D intercepts YB and transmits YD2 to „A" and „D" calculate K1 $K1 = (Y_B)^{X_{D1}} \bmod q$
7. A receives YD2 and calculates $K2 = (Y_{D2})^{X_A} \bmod q$

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key *K*1 and Alice and Darth share secret key *K*2. All future communication between Bob and Alice is compromised in the following way.

1. A sends an encrypted message *M*: E(*K*2, *M*).
2. D intercepts the encrypted message and decrypts it to recover *M*.
3. D sends B *E*(*K*1, *M*) or E(*K*1, *M*′), where *M*′ is any message. In the first case, D simply wants to eavesdrop on the communication without altering it. In the second case, D wants to modify the message going to B

The key exchange protocol is vulnerable to such an attack because it does not authenticate the

participants. This vulnerability can be overcome with the use of digital signatures and public-key certificates.

UNIT-III

HASH FUNCTION:

It is a one of the authentication function; it accepts a variable size message M as input and produces a fixed size output.

A hash value ‘h’ is generated by a function H of the form

h=H (M)

M variable length message H(M)

fixed length hash value.

The hash code is also referred as Message Digest (MD) or hash value.

The main difference between Hash Function and MAC is a hash code does not use a key but is a function only of the input message.

The hash value is appended to the message at the source at a time when the message is assumed or known to be correct.

The receiver authenticates that message by re-computing the hash value.

Hash functions are often used to determine whether or not data has changed.

Figure 11.1 depicts the general operation of a cryptographic hash function

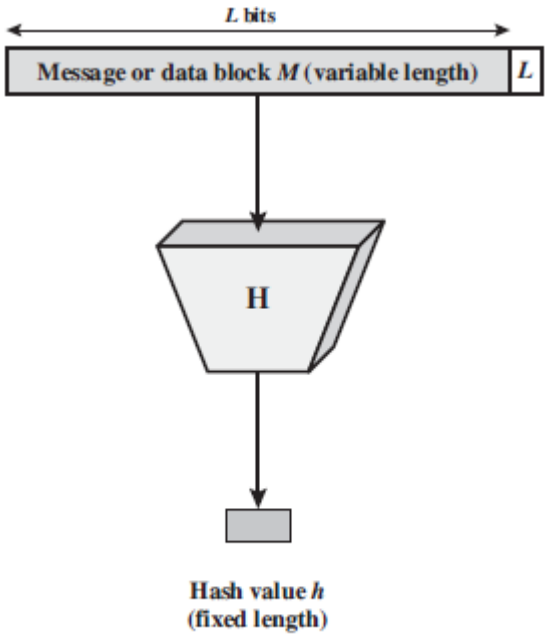


Figure 11.1 Black Diagram of Cryptographic Hash Function; $h = H(M)$

APPLICATIONS OF CRYPTOGRAPHIC HASH FUNCTIONS

It is used in a wide variety of security applications and Internet protocols

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 (i.e., contain no modification, insertion, deletion, or replay)

When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest.

Figure 11.2 illustrates a variety of ways in which a hash code can be used to provide message authentication, as follows.

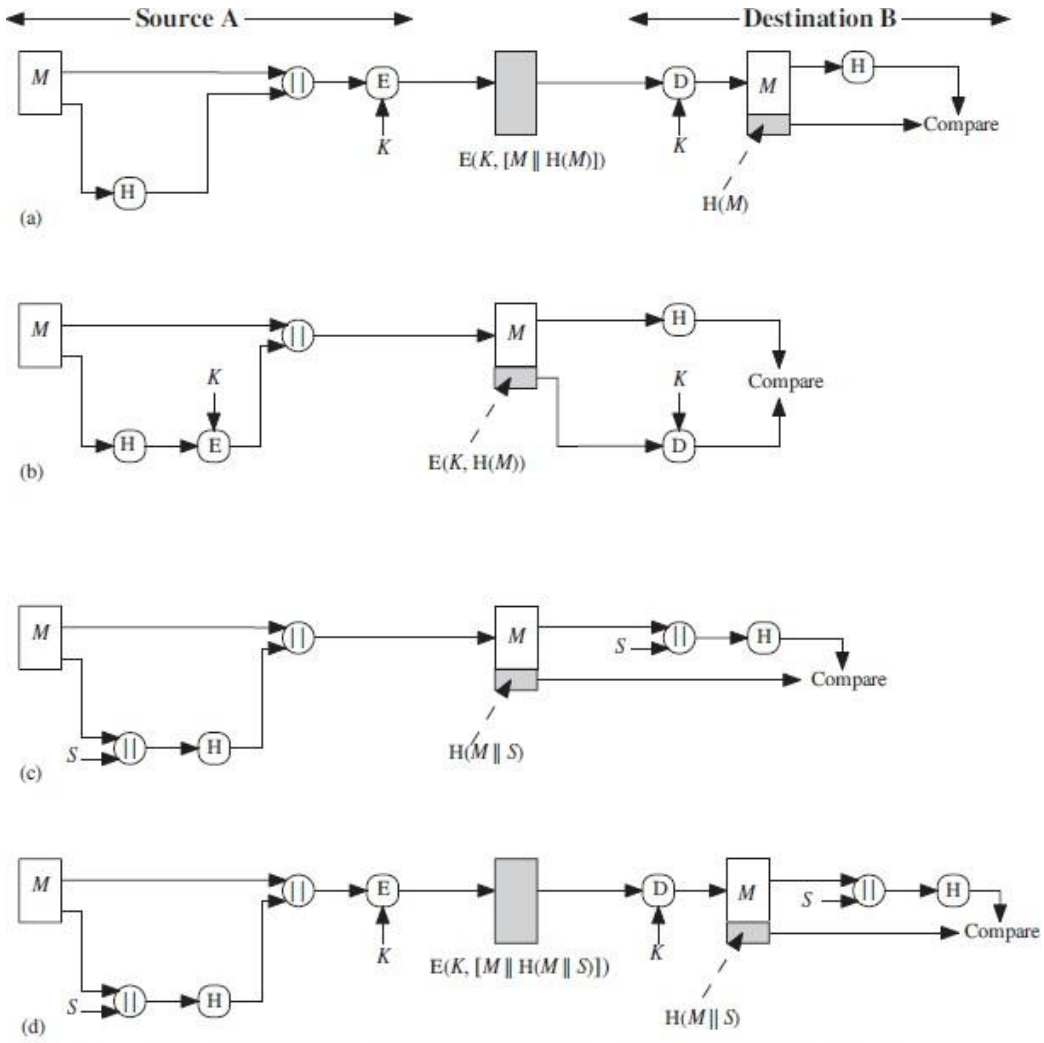


Figure 11.2 Simplified Examples of the Use of a Hash Function for Message Authentication

(a) The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered.

The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

(b) Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality

(c) It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value S.A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses, it can recomputed the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

(d) Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code.

Digital Signatures

Another important application, which is similar to the message authentication application, is the digital signature.

The operation of the digital signature is similar to that of the MAC. In the case of the digital signature, the hash value of a message is encrypted with a user's private key. Anyone who knows the user's public key can verify the integrity of the message that is associated with the digital signature.

Figure 11.3 illustrates, in a simplified fashion, how a hash code is used to provide a digital signature.

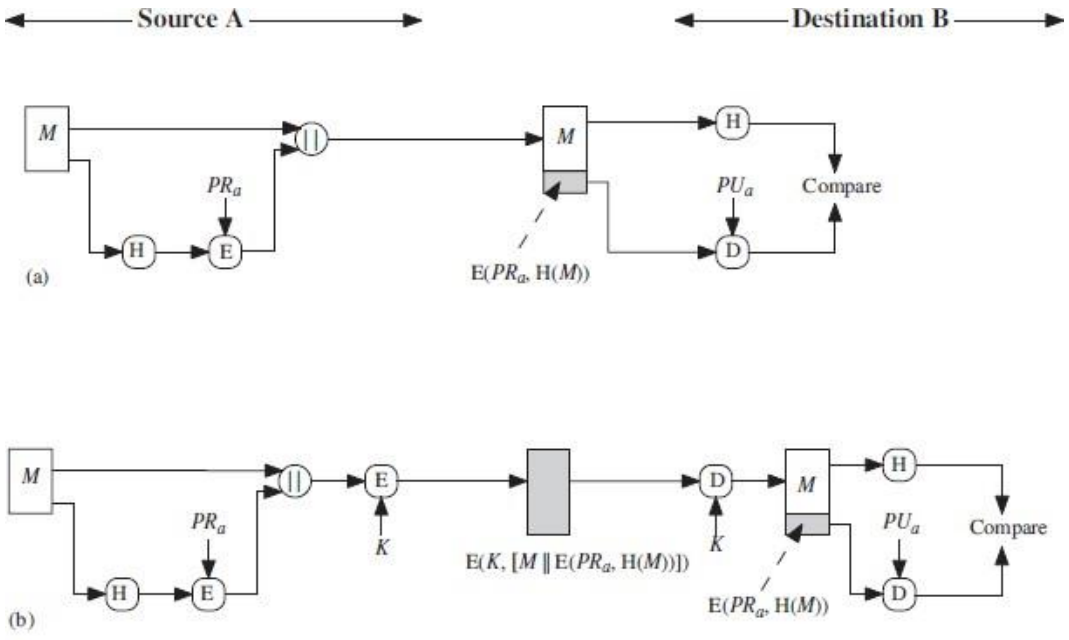


Figure 11.3 Simplified Examples of Digital Signatures

REQUIREMENTS& SECURITY FOR A HASH FUNCTION:

The purpose of a hash function is to produce a "fingerprint" of a file, message or other block of data. To be useful for message authentication, a hash function H must have the following properties:

H can be applied to a block of data of any size

H produces a fixed length output.

H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.

One-way property: - for any given value h, it is computationally infeasible to find x such that H(x)=h. this sometimes referred to in the literature as the one way property.

Weak collision resistance:- for any given block x. it is computationally infeasible to find y≠x with H(y)=H(x). this is referred as weak collision resistance.

Strong collision resistance:- it is computationally infeasible to find any pair (X,Y) such that H(x)=H(y). this is referred as strong collision resistance.

Requirements for a Cryptographic Hash Function H

Requirement	Description
Variable input size	H can be applied to a block of data of any size.
Fixed output size	H produces a fixed-length output.
Efficiency	$H(x)$ is relatively easy to compute for any given x, making both hardware and software implementations practical.
Preimage resistant (one-way property)	For any given hash value h, it is computationally infeasible to find y such that $H(y) = h$.
Second preimage resistant (weak collision resistant)	For any given block x, it is computationally infeasible to find $y \neq x$ with $H(y) = H(x)$.
Collision resistant (strong collision resistant)	It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.
Pseudorandomness	Output of H meets standard tests for pseudorandomness.

A hash function that satisfies the first five properties in Table 11.1 is referred to as a weak hash function. If the sixth property, collision resistant, is also satisfied, then it is referred to as a strong hash function.

As with encryption algorithms, there are two categories of attacks on hash functions: brute-force attacks and cryptanalysis

Brute-Force Attacks

A brute-force attack does not depend on the specific algorithm but depends only on bit length. In the case of a hash function, a brute-force attack depends only on the bit length of the hash value. A cryptanalysis, in contrast, is an attack based on weaknesses in a particular cryptographic algorithm.

Cryptanalysis

As with encryption algorithms, cryptanalytic attacks on hash functions seek to exploit some property of the algorithm to perform some attack other than an exhaustive search. The way to measure the resistance of a hash algorithm to cryptanalysis is to compare its strength to the effort required for a brute-force attack.

That is, an ideal hash algorithm will require a cryptanalytic effort greater than or equal to the brute-force effort.

SHA(Secure Hash Algorithm):

In recent years, the most widely used hash function has been the Secure Hash Algorithm (SHA).

Introduction:

The Secure Hash Algorithm is a family of cryptographic hash functions developed by the NIST (National Institute of Standards & Technology).

SHA is based on the MD4 algorithm and its design closely models MD5.

SHA-1 is specified in RFC 3174.

Purpose: Authentication, not encryption.

SHA-1 produces a hash value of 160 bits. In 2002, NIST produced a revised version of the standard, FIPS 180-2, that defined three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512, respectively.

SHA-1 logic:

The algorithm takes a message with maximum of length of less than 264 bits.

Produce output is 160 bits message digest.

The input is processed 512 bits block.

Processed Steps:

Algorithm processing Steps:

Step1: Append Padding Bits

Step 2: Append Length

Step 3: Initialize MD Buffer

Step 4: Process Message in 512 bit (16-Word) Blocks

Step 5: Output

Table 11.3 Comparison of SHA Parameters

	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Message Digest Size	160	224	256	384	512
Message Size	$< 2^{64}$	$< 2^{64}$	$< 2^{64}$	$< 2^{128}$	$< 2^{128}$
Block Size	512	512	512	1024	1024
Word Size	32	32	32	64	64
Number of Steps	80	64	64	80	80

Note: All sizes are measured in bits.

SHA-512 Logic

The algorithm takes as input a message with a maximum length of less than 2128 bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks.

This follows the general structure depicted in Figure 11.8. The processing consists of the following steps.

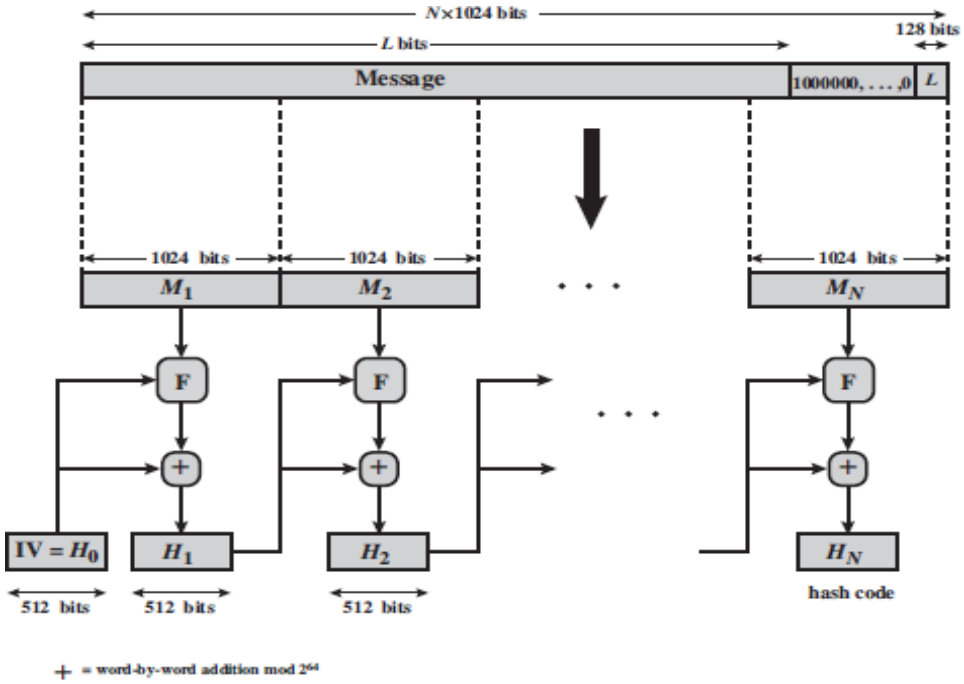


Figure 11.8 Message Digest Generation Using SHA-512

Step 1 Append padding bits.

The message is padded so that its length is congruent to 896 modulo 1024 [length K 896(mod 1024)]. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 1024. The padding consists of a single 1 bit followed by the necessary number of 0 bits.

Step 2 Append lengths.

A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the

padding).

The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. In Figure 11.9, the expanded message is represented as the sequence of 1024-bit blocks M1, M2, c, MN, so that the total length of the expanded message is N * 1024bits.

Step 3 Initialize hash buffer.

A 512-bit buffer is used to hold intermediate and final results of the hash function. The buffer can be represented as eight 64-bit registers (a, b, c, d, e, f, g, h). These registers are initialized to the following 64-bit integers (hexadecimal values):

a = 6A09E667F3BCC908 e = 510E527FADE682D1

b = BB67AE8584CAA73B f = 9B05688C2B3E6C1F

c = 3C6EF372FE94F82B g = 1F83D9ABFB41BD6B

d = A54FF53A5F1D36F1 h = 5BE0CD19137E2179

These values are stored in big-endian format, which is the most significant byte of a word in the low-address (leftmost) byte position. These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

Step 4 Process message in 1024-bit (128-word) blocks.

The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F.

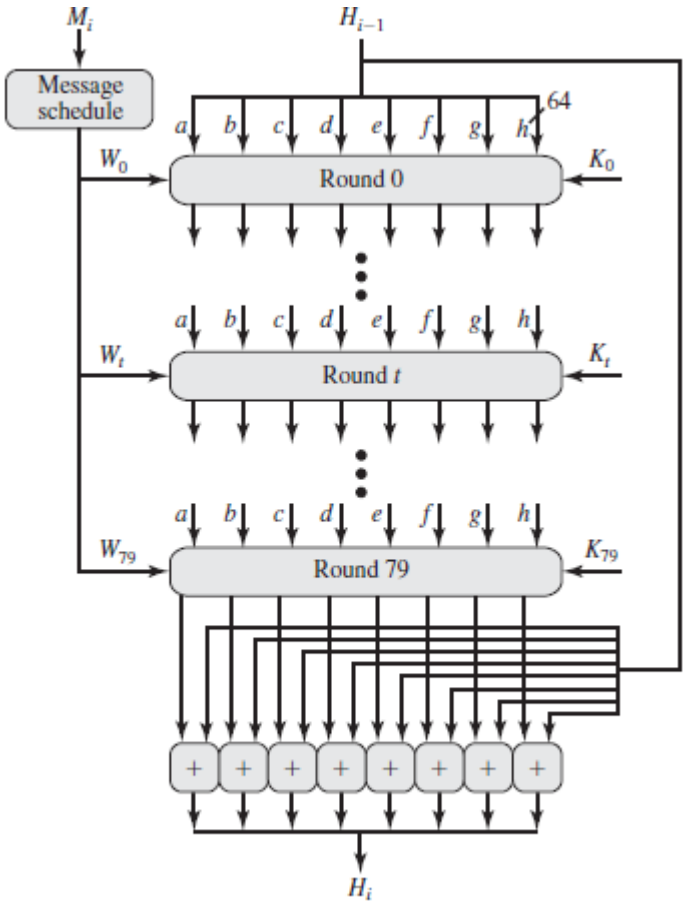


Figure 11.9 SHA-512 Processing of a Single 1024-Bit Block

Each round takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer. At input to the first round, the buffer has the value of the intermediate hash value, Hi-1. Each round t makes use of a 64-bit value Wt, derived from the current 1024-bit block being processed (Mi). These values are derived using a message schedule described subsequently. Each round also makes use of an additive constant Kt, where 0 … t … 79 indicates one of the 80 rounds. These words represent the first 64 bits of the fractional parts of the cube roots of the first 80 prime numbers. The constants provide a "randomized" set of 64-bit patterns, which should eliminate any regularities in the input data. Table 11.4 shows these constants in hexadecimal format (from left to right).
The output of the eightieth round is added to the input to the first round (Hi-1) to produce Hi. The addition is done independently for each of the eight words in the buffer with each of the corresponding words in Hi-1, using addition modulo 264. Step 5 Output. After all N 1024-bit blocks have been processed, the output from the Nth stage is the 512-bit message digest

Step 5 Output. After all N 1024-bit blocks have been processed, the output from the Nth stage is the 512-bit message digest.

We can summarize the behavior of SHA-512 as follows:

$$H_0 = \text{IV}$$
$$H_i = \text{SUM}_{64}(H_{i-1}, \text{abcdefgh}_i)$$
$$MD = H_N$$

where

IV = initial value of the abcdefgh buffer, defined in step 3

abcdefgh_i = the output of the last round of processing of the ith message block

N = the number of blocks in the message (including padding and length fields)

SUM_{64} = addition modulo 2^{64} performed separately on each word of the pair of inputs

MD = final message digest value

It remains to indicate how the 64-bit word values W_t are derived from the 1024-bit message. Figure 11.12 illustrates the mapping. The first 16 values of W_t are taken directly from the 16 words of the current block. The remaining values are defined as

$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

where

$$\begin{aligned}
\sigma_0^{512}(x) &= \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x) \\
\sigma_1^{512}(x) &= \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x) \\
\text{ROTR}^n(x) &= \text{circular right shift (rotation) of the 64-bit argument } x \text{ by } n \text{ bits} \\
\text{SHR}^n(x) &= \text{left shift of the 64-bit argument } x \text{ by } n \text{ bits with padding by zeros on the right} \\
+ &= \text{addition modulo } 2^{64}
\end{aligned}$$

Thus, in the first 16 steps of processing, the value of W_t is equal to the corresponding word in the message block. For the remaining 64 steps, the value of W_t consists of the circular left shift by one bit of the XOR of four of the preceding values of W_t, with two of those values subjected to shift and rotate operations. This introduces a great deal of redundancy and interdependence into the message

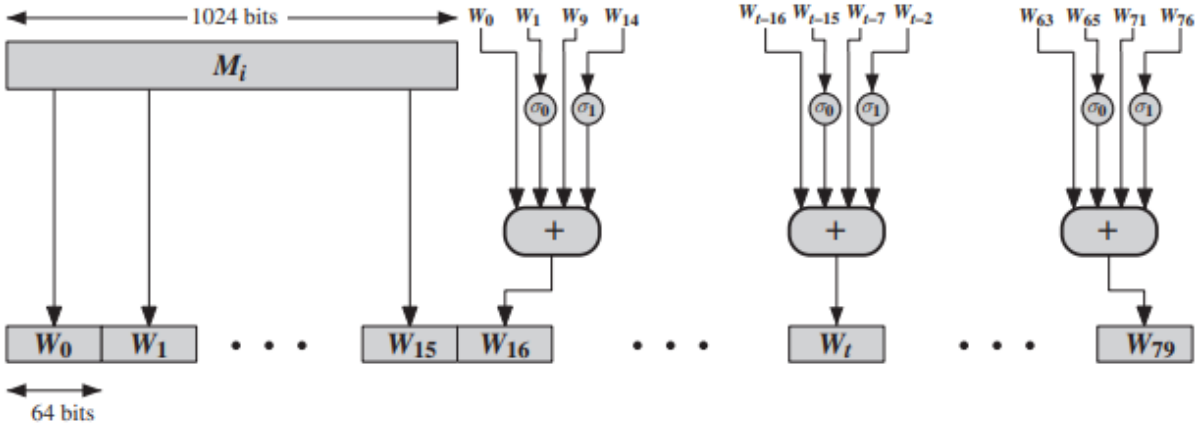


Figure 11.12 Creation of 80-word Input Sequence for SHA-512 Processing of Single Block

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.

MESSAGE AUTHENTICATION REQUIREMENTS

In the context of communications across a network, the following attacks can be identified

1. Disclosure: Release of message contents to any person or process not possessing the appropriate cryptographic key.

2. Traffic analysis: Discovery of the pattern of traffic between parties. In a connection oriented application, the frequency and duration of connections could be determined.

3. Masquerade: Insertion of messages into the network from a fraudulent source.

4. Content modification: Changes to the contents of a message, including insertion, deletion, transposition, and modification.

5. Sequence modification: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

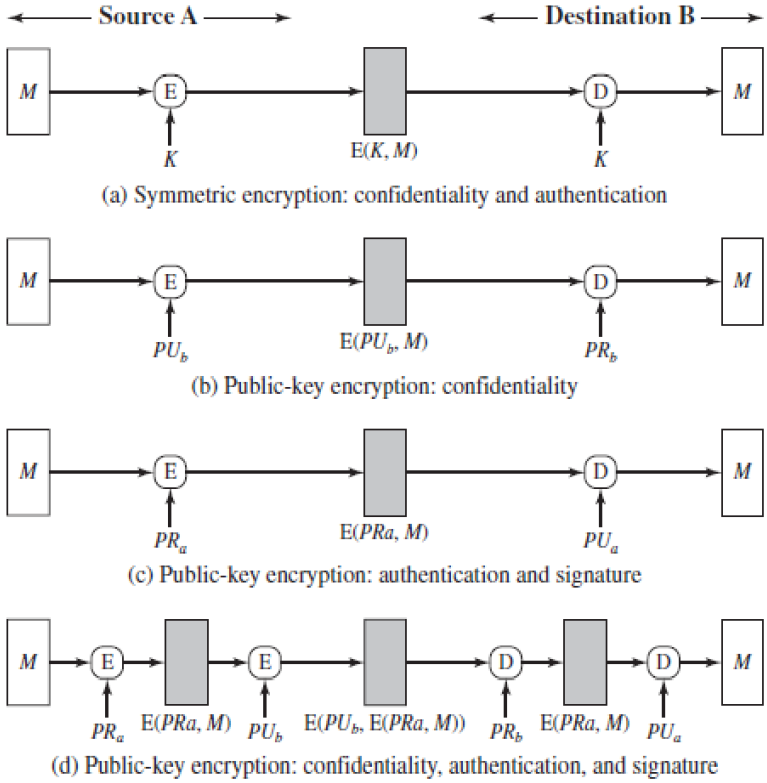
6. Timing modification: Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed.

7. Source repudiation: Denial of transmission of message by source.

8. Destination repudiation: Denial of receipt of message by destination.

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



(a) Symmetric encryption: confidentiality and authentication

(b) Public-key encryption: confidentiality

(c) Public-key encryption: authentication and signature

(d) Public-key encryption: confidentiality, authentication, and signature

Figure 12.1 Basic Uses of Message Encryption

---

authentication protocol that enables a receiver to verify the authenticity of a message. there are 3 types of functions that may be used to produce an authenticator.

- **Hash function:** A function that maps a message of any length into a fixed length hash value, which serves as the authenticator
- **Message encryption:** The cipher text of the entire message serves as its authenticator
- **Message authentication code (MAC):** A function of the message and a secret key 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.

MESSAGE AUTHENTICATION CODE (MAC)

This authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a **cryptographic checksum** or MAC, that is appended to the message. This

technique assumes that two communicating parties, say A and B, share a common secret key

When A has a message to send to B, it calculates the MAC as a function of the message and the key

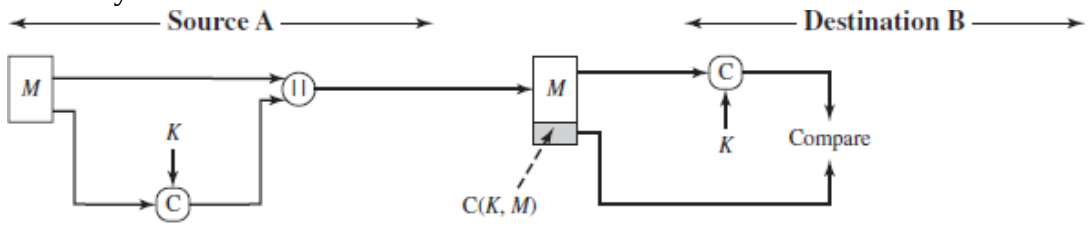
$$MAC = MAC(K, M)$$

where

M = input message
C = MAC function
K = shared secret key
MAC = message authentication code

The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The receivedMAC is compared to the calculated MAC (Figure 12.4a). If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key.



(a) Message authentication

SECURITY OF MACS:

Just as with symmetric and public-key encryption, we can group attacks on hash functions and MACs into two categories: **brute-force attacks and cryptanalysis.**

brute-force attacks

A brute-force attack on a MAC is a more difficult undertaking than a brute-force attack on a hash function because it requires known message-tag pairs. The strength of a hash function against brute-force attacks depends solely on the length of the hash code produced by the algorithm, with cost (2m/2). A brute-force attack on a MAC has cost related to min(2k, 2n), similar to symmetric encryption algorithms. It would appear reasonable to require that the key length and MAC length satisfy a relationship such as min(k, n) >= N, where N is perhaps in the range of 128 bits.

cryptanalysis.

As with encryption algorithms, cryptanalytic attacks on hash functions and MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search. The way to measure the resistance of a hash or MAC algorithm to cryptanalysis is to compare its strength to the effort required for a brute force attack. That is, an ideal hash or MAC algorithm willrequire a cryptanalytic effort greater than or equal to the brute-force effort.

HMAC:

In recent years, there has been increased interest in developing a MAC derived from a cryptographic hash function, because they generally execute faster than symmetric block ciphers, and because code for cryptographic hash functions is widely available.

A hash function such as SHA was not designed for use as a MAC and cannot be used directly for that purpose because it does not rely on a secret key. There have been a number of proposals for the incorporation of a secret key into an existing hash algorithm, originally by just pre-pending a key to the message. Problems were found with these earlier, simpler proposals, but they resulted in the development of HMAC.

HMAC Design Objectives:

• To use, without modifications, available hash functions. In particular, to use hash functions that perform well in software and for which code is freely and widely available.

• To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.

• To preserve the original performance of the hash function without incurring a significant degradation.

• To use and handle keys in a simple way.

• To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions about the embedded hash function.

HMAC Algorithm:

HMAC Algorithm

Figure 12.5 illustrates the overall operation of HMAC. Define the following terms.

H = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)

IV = initial value input to hash function

M = message input to HMAC (including the padding specified in the embedded hash function)

Y_i = i th block of $M, 0 \leq i \leq (L-1)$

L = number of blocks in M

b = number of bits in a block

n = length of hash code produced by embedded hash function

K = secret key; recommended length is $\geq n$; if key length is greater than b, the key is input to the hash function to produce an n-bit key

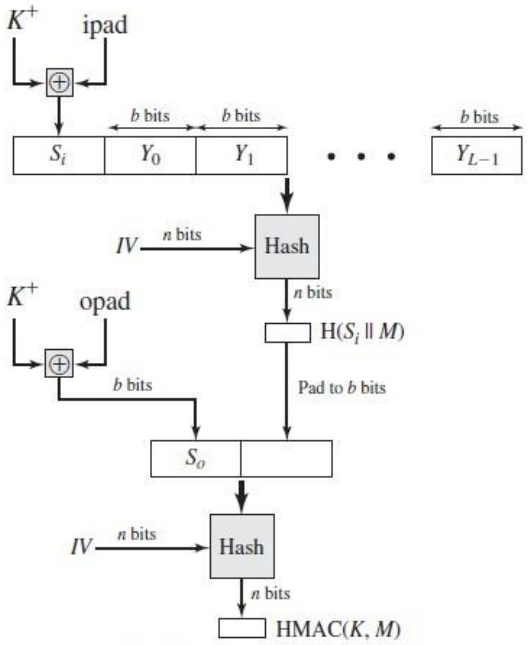
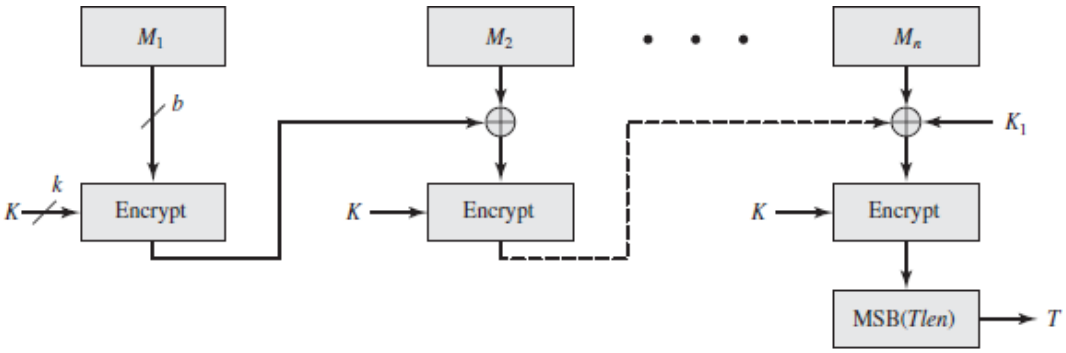


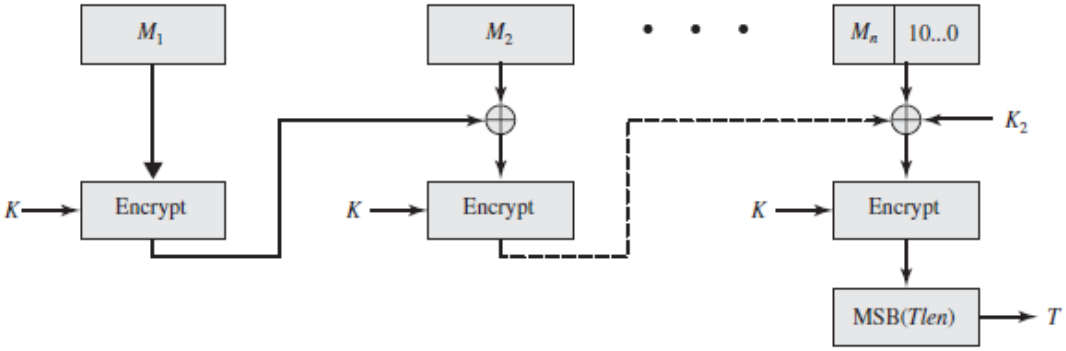
Figure 12.5 HMAC Structure

Cipher-Based Message Authentication Code (CMAC)

First, let us define the operation of CMAC when the message is an integer multiple n of the cipher block length b. For AES, $b = 128$, and for triple DES, $b = 64$. The message is divided into n blocks $(M_1, M_2, \ldots, M_n)$. The algorithm makes use of a k-bit encryption key K and an n-bit constant, K_1. For AES, the key size k is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits. CMAC is calculated as follows (Figure 12.8).



(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

Figure 12.8 Cipher-Based Message Authentication Code (CMAC)

$$
\begin{aligned}
C_1 &= \mathrm{E}(K, M_1) \\
C_2 &= \mathrm{E}(K, [M_2 \oplus C_1]) \\
C_3 &= \mathrm{E}(K, [M_3 \oplus C_2]) \\
&\bullet \\
&\bullet \\
&\bullet \\
C_n &= \mathrm{E}(K, [M_n \oplus C_{n-1} \oplus K_1]) \\
T &= \mathrm{MSB}_{Tlen}(C_n)
\end{aligned}
$$

where

T = message authetication code, also referred to as the tag
$Tlen$ = bit length of T
$\mathrm{MSB}_s(X)$ = the s leftmost bits of the bit string X

If the message is not an integer multiple of the cipher block length, then the final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary so that the final block is also of length b. The CMAC operation then proceeds as before, except that a different n-bit key K_2 is used instead of K_1.

DIGITAL SIGNATURES

A digital signature is an authentication mechanism that enables the creator of a message to attach acode 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 source andintegrity of the message.

The digital signature standard (DSS) is an NIST standard that uses the secure hash algorithm (SHA).

Properties

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.

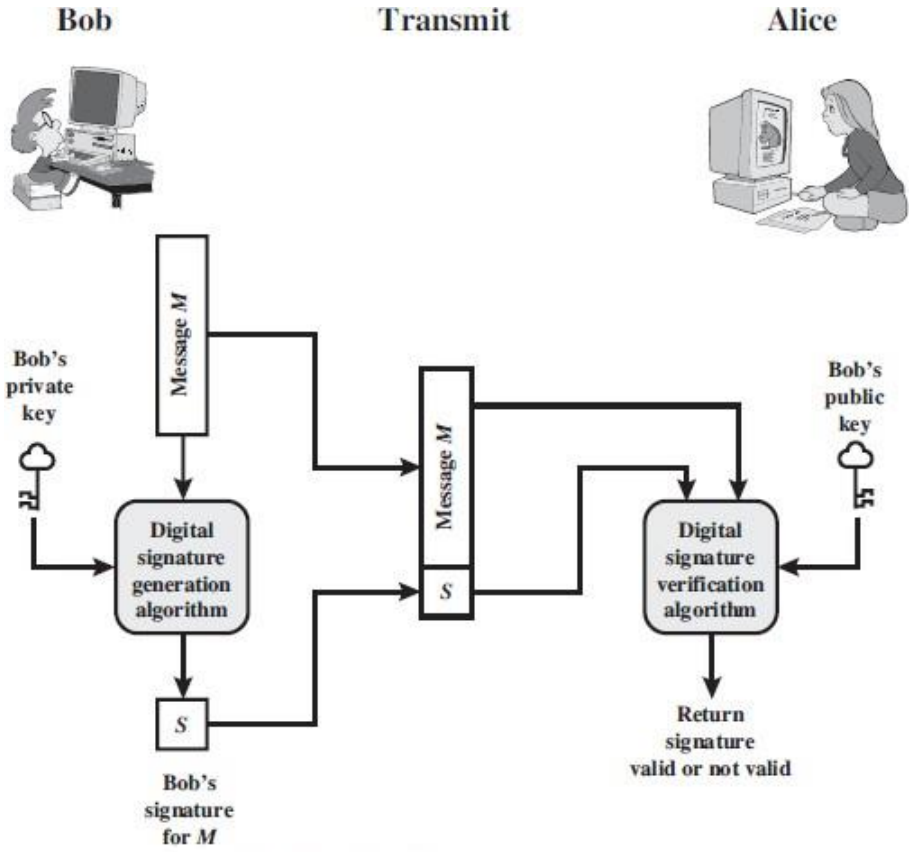


Figure 13.1 Generic Model of Digital Signature Process

DIGITAL SIGNATURE STANDARD

The Digital Signature Standard (DSS) makes use of the Secure Hash Algorithm (SHA) described and presents a new digital signature technique, the Digital Signature

Algorithm (DSA).

This latest version incorporates digital signature algorithms based on RSA and on elliptic curve cryptography. In this section, we discuss the original DSS algorithm. The DSS uses an algorithm that is designed to provide only the digital signature function.Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique.

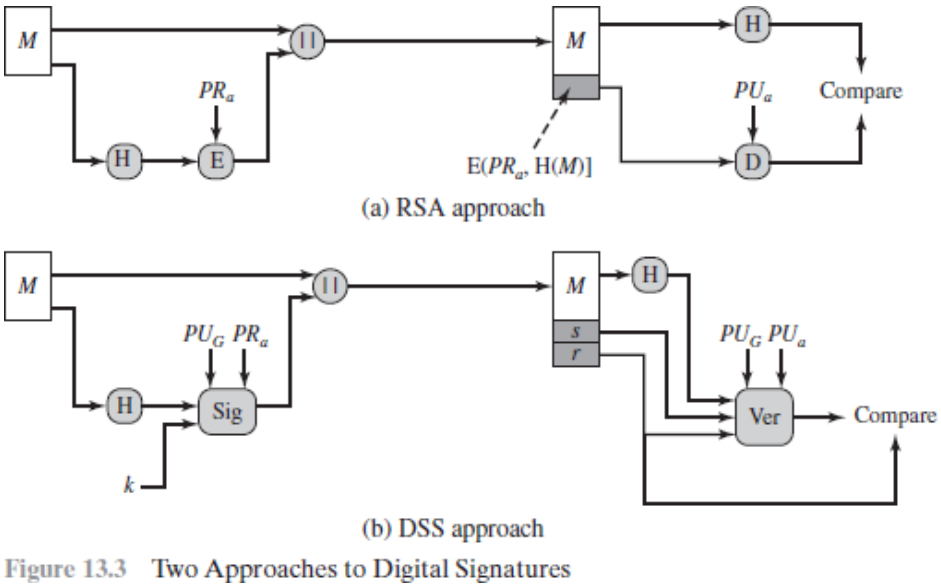


Figure 13.3 Two Approaches to Digital Signatures

In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code.

The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.

Digital Signature Algorithm

The DSA is based on the difficulty of computing discrete logarithms and is based on schemes originally presented by Elgamal and Schnorr. The DSA signature scheme has advantages, being both smaller (320 vs 1024bit) and faster over RSA. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique

DSA typically uses a common set of global parameters (p,q,g) for a community of clients, as shown. A 160-bit prime number q is chosen. Next, a prime number p is selected with a length between 512 and 1024 bits such that q divides (p – 1). Finally, g is chosen to be of the form $h^{(p-1)/q}$ *mod p* where h is an integer between 1 and (p – 1) with the restriction that g must be greater than 1.Thus, the global public key components of DSA have the same for as in the Schnorr signature scheme.

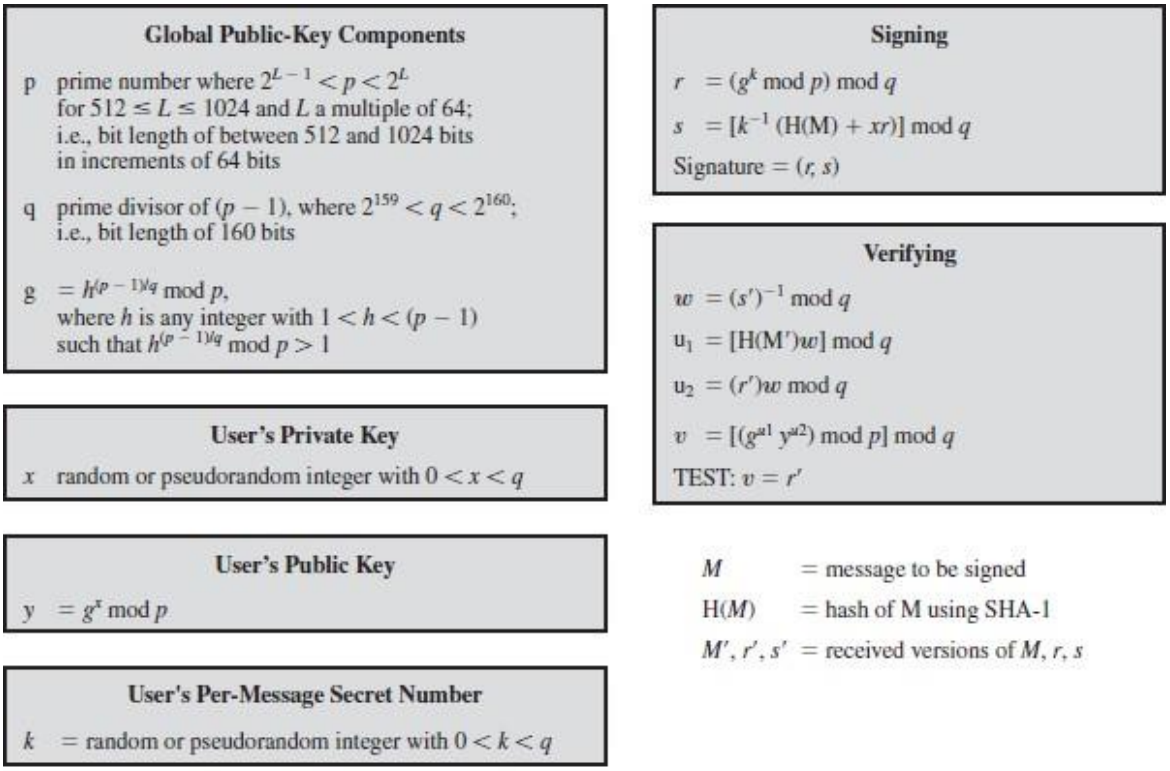
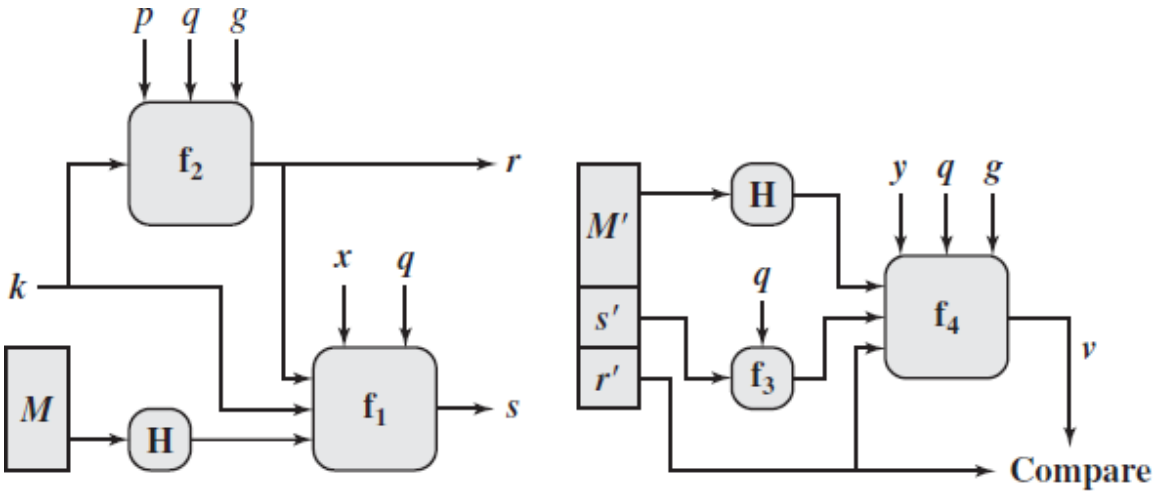


Figure 13.4 The Digital Signature Algorithm (DSA)

Signing and Verifying

The structure of the algorithm, as revealed here is quite interesting. Note that the test at the end is on the value r, which does not depend on the message at all. Instead, r is a function of k and the three global public-key components. The multiplicative inverse of k (mod q) is passed to a function that also has as inputs the message hash code and the user's private key. The structure of this function is such that the receiver can recover r using the incoming message and signature, the public key of the user, and the global public key.



$s = f_1(H(M), k, x, r, q) = (k^{-1} (H(M) + xr)) \bmod q$

$r = f_2(k, p, q, g) = (g^k \bmod p) \bmod q$

(a) Signing

$w = f_3(s', q) = (s')^{-1} \bmod q$

$v = f_4(y, q, g, H(M'), w, r')$

$= ((g^{(H(M')w) \bmod q} y^{r'w \bmod q}) \bmod p) \bmod q$

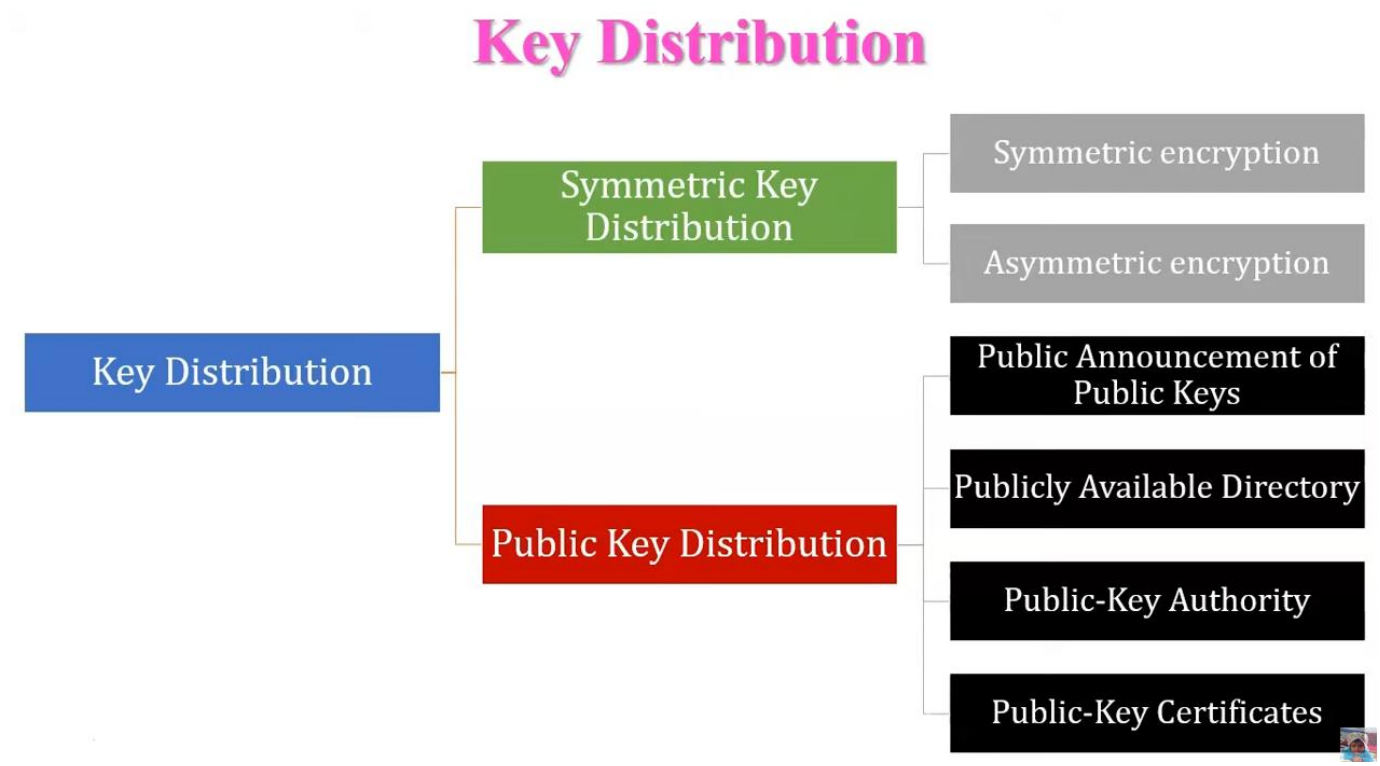
(b) Verifying

Figure 13.5 DSS Signing and Verifying

KEY MANAGEMENT AND DISTRIBUTION

Key Management

- The main aim of key management is to generate a secret key between two parties and store it to prove the authenticity between communicating users.
- Key management is the techniques which support ***key generation, storage*** and ***maintenance of the key*** between authorized users.
- Key management plays an important role in cryptography as the basis for securing cryptographic goals like ***confidentiality, authentication, data integrity*** and ***digital signatures.***
- It is not the case where communicating parties are using same key for encryption and decryption or whether two different keys are used for encryption and decryption.
- Basic purpose of key management is ***key generation, key distribution, controlling the use of keys, updating, destruction of keys and key backup/recovery.***



Key management and Distribution Symmetric Key Distribution Using Symmetric Encryption

For **symmetric** encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Therefore, the term that refers to the means of delivering a key to two parties who wish to exchange data, without allowing others to see the key. For two parties A and B, key distribution can be achieved in anumber of ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the newkey to the other, encrypted using the old key.
4. If A and B each has an encrypted connection to a third party C, C can deliver a key onthe encrypted links to A and B.

Physical delivery (1 & 2) is simplest - but only applicable when there is personal contact between recipient and key issuer. This is fine for link encryption where devices & keys occur in pairs, but does not scale as number of parties who wish to communicate grows.3 is mostly based on 1 or 2 occurring first.

A third party, whom all parties trust, can be used as a **trusted intermediary** to mediate the establishment of secure communications between them (4). Must trust intermediary not to abuse the knowledge of all session keys. As number of parties grow, some variant of 4 is only practical solution to the huge growth in number of keys potentially needed.

Key distribution centre:

- The use of a **key distribution center** is based on the use of a hierarchy of keys. At a minimum, two levels of keys are used.
- Communication between end systems is encrypted using a temporary key, often referredto as a **Session key**.
- Typically, the session key is used for the duration of a logical connection and then discarded
- **Master key** is shared by the key distribution center and an end system or user and used toencrypt the session key.

Key Distribution Scenario:

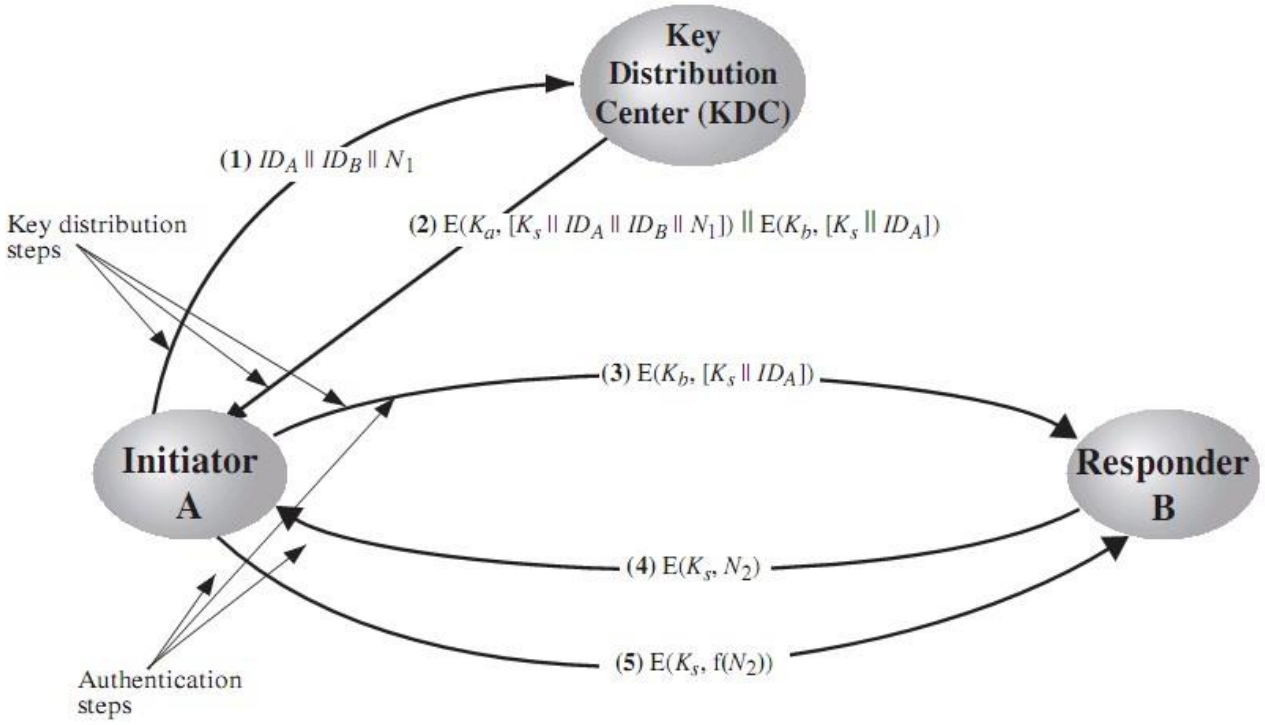


Figure 14.3 Key Distribution Scenario

Let us assume that user A wishes to establish a logical connection with B and requiresa one-

time session key to protect the data transmitted over the connection. A has a master key, K_a, known only to itself and the KDC; similarly, B shares the master key K_b with the KDC. The following steps occur:

1 A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N_1, for this transaction, which we refer to as a **nonce**. The nonce may be a timestamp, a counter, or a random number; the minimum requirement is **that it differs with each request.** Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.

2. The KDC responds with a message encrypted using Ka Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:

- The **one-time session key, Ks,** to be used for the session
- The **original request message,** including the nonce, to enable A to match this response with the appropriate request

Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request. In addition, the message includes two items intended for B:

- The one-time session key, Ks to be used for the session
- An identifier of A (e.g., its network address), IDA

These last two items are encrypted with Kb (the master key that the KDC shares with B). They are to be sent to B to establish the connection and prove A's identity.

3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely, $E(K_b, [K_s \| ID_A])$. Because this information is encrypted with K_b, it is protected from eavesdropping. B now knows the session key (K_s), knows that the other party is A (from ID_A), and knows that the information originated at the KDC (because it is encrypted using K_b). At this point, a session key has been securely delivered to A and B, and they may begin

their protected exchange. However, two additional steps are desirable:

4. Using the newly minted session key for encryption, B sends a nonce, N_2, to A.
5. Also using K_s, A responds with $f(N_2)$, where f is a function that performs sometransformation on N_2 (e.g., adding one).

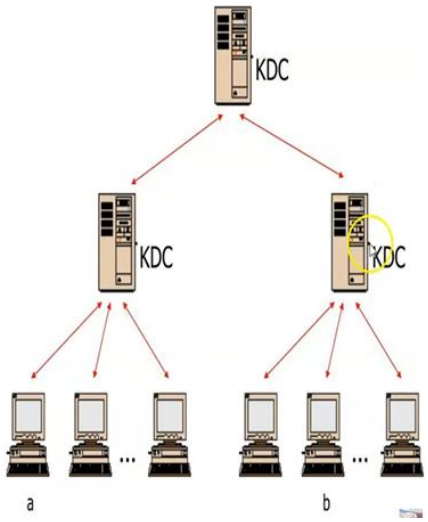
These steps assure B that the original message it received (step 3) was not a replay.

Note that the actual key distribution involves only steps 1 through 3 but that steps 4 and 5, aswell as 3, perform an authentication function.

Major Issues with KDC:

Hierarchical Key Control

- It is not **necessary to limit** the key distribution function to a single KDC.Indeed,for very large networks,it may not be practical to do so.As an alternative,a hierarchy of KDCs can be established.
- For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building.
- If two entities in different domains desire a shared key, then the corresponding localKDCs can communicate through a global KDC.



- The hierarchical concept can be extended to three or even more layers, depending on thesize of the user population and the geographic scope of the internetwork.
- A hierarchical scheme **minimizes the effort involved in master key distribution**,because most

master keys are those shared by a local KDC with its local entities.

Session Key Lifetime

- The distribution of session keys delays the start of any exchange and places a burden on network capacity. A security manager must try to balance these competing considerationsin determining the lifetime of a particular session key.
- For **connection-oriented protocols**, one obvious choice is to use the same session key for the length of time that the connection is open, using a new session key for each newsession.
- If a logical connection has a very long lifetime, then it would be prudent to change thesession key periodically, perhaps every time the PDU (protocol data unit) sequence number cycles.
- For a **connectionless protocol**, such as a transaction-oriented protocol, there is noexplicit connection initiation or termination.
- Thus, it is not obvious how often one needs to change the session key. The most secure approach is to use a **new session key for each exchange**.
- A better strategy is to use a given **session key for a certain fixed period only or for acertain number of transactions.**

A Transparent Key Control Scheme

- The approach suggested in Figure 14.3is useful for providing end-to-end encryption at a network or transport level in a way that is transparent to the end users.
- The approach assumes that communication makes use of a connection-oriented end-to- end protocol, such as TCP.
- The noteworthy element of this approach is a session security module (SSM), which may consist of functionality at one protocol layer,that performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

The steps involved in establishing a connection are shown in Figure

1. When one host wishes to set up a connection to another host, it transmits a connection-request packet.
2. The SSM saves that packet and applies to the KDC for permission to establishthe connection.

3. The communication between the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC.If the KDC approves the connection request,it generates the session key and delivers it to the two appropriate SSMs,using a unique permanent key for each SSM.
4. The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems.
5. All user data exchanged between the two end systems are encrypted by their respective SSMs using the onetime session key.

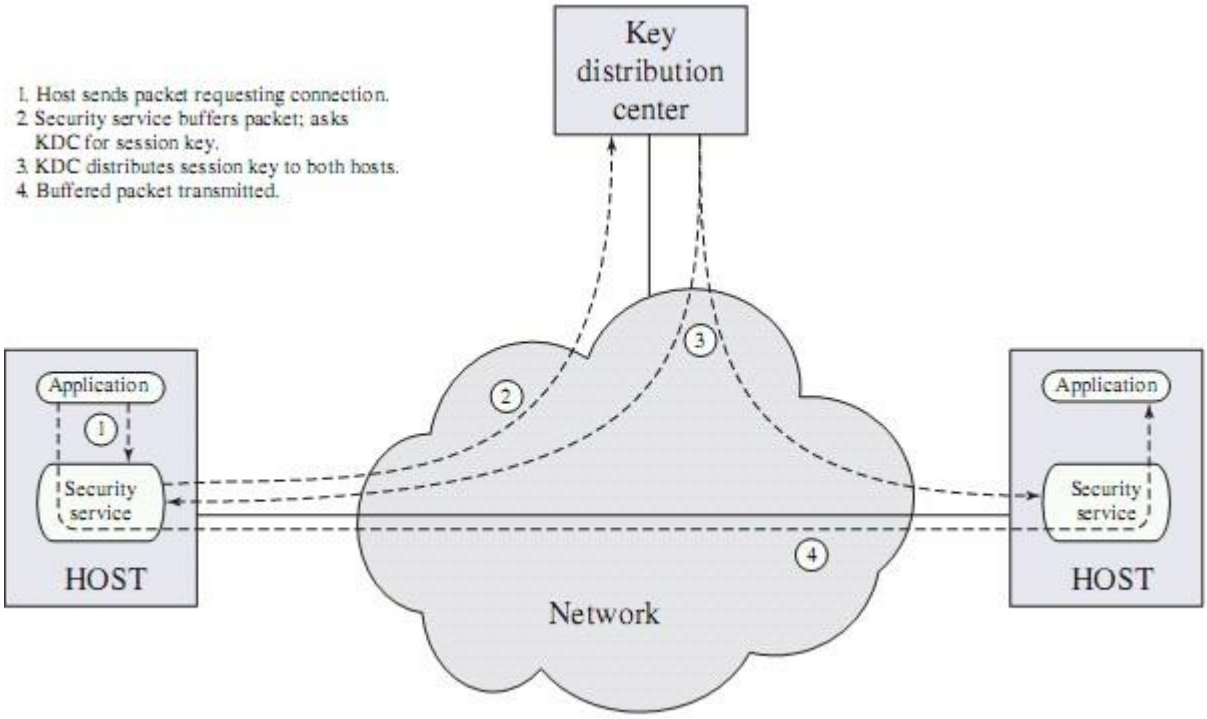


Figure 14.4 Automatic Key Distribution for Connection-Oriented Protocol

- The automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other.

Decentralized Key Control

- The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distributionis fully decentralized.

- Although full decentralization is not practical for larger networks using symmetric encryption only, it may be useful within a local context.
- A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution.
- Thus, there may need to be as many as $(n - 1)/2$ master keys for a configuration with n end systems.
- A session key may be established with the following sequence of steps (Figure 14.5).

1. A issues a request to B for a session key and includes a nonce, .

2. B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B,an identifier of B,the value f(N_1), andanother nonce N_2.

3. Using the new session key,A returns f(N_2) to B.

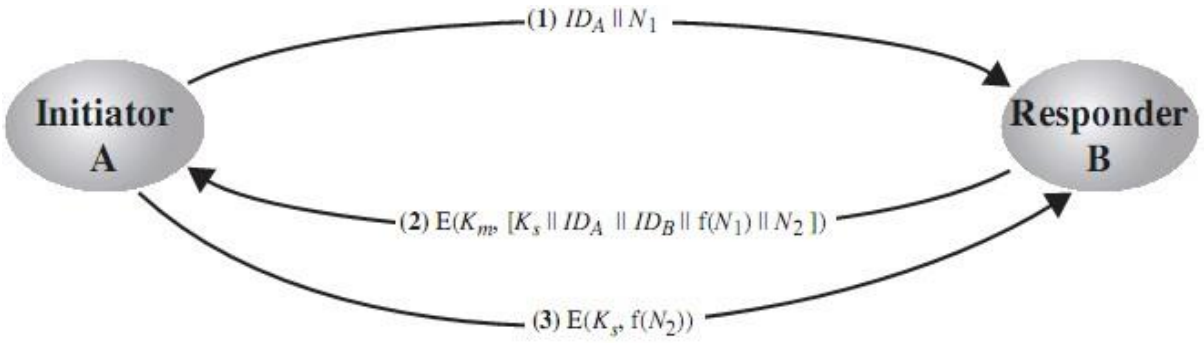


Figure 14.5 Decentralized Key Distribution

Controlling Key Usage

The concept of a key hierarchy and the use of automated key distribution techniques greatly reduce the number of keys that must be manually managed and distributed. It also may be desirable to impose some control on the way in which automatically distributed keys are used. For example, in addition to separating master keys from session keys, we may wish to define different types of session keys on the basis of use, such as

- Data-encrypting key, for general communication across a network
- PIN-encrypting key, for personal identification numbers (PINs) used in electronic funds transfer and point-of-sale applications
- File-encrypting key, for encrypting files stored in publicly accessible locations

To illustrate the value of separating keys by type, consider the risk that a master key is imported as a data-encrypting key into a device. Normally, the master key is physically secured within the cryptographic hardware of the key distribution center and of the end systems. Session keys encrypted with this master key are available to application programs, as are the data encrypted with such session keys.

However, if a master key is treated as a session key, it may be possible for an unauthorized application to obtain plaintext of session keys encrypted with that master key.

The proposed technique is for use with DES and makes use of the extra 8 bits in each64-bit DES key. That is, the eight non-key bits ordinarily reserved for parity checking form the key tag. The bits have the following interpretation:

- One bit indicates whether the key is a session key or a master key.
- One bit indicates whether the key can be used for encryption.
- One bit indicates whether the key can be used for decryption.
- The remaining bits are spares for future use.

Because the tag is embedded in the key, it is encrypted along with the key when thatkey is distributed, thus providing protection. The drawbacks of this scheme are

1. The tag length is limited to 8 bits, limiting its flexibility and functionality.
2. Because the tag is not transmitted in clear form, it can be used only at thepoint of decryption, limiting the ways in which key use can be controlled.

A more flexible scheme, referred to as the control vector, is described here. In this scheme, each session key has an associated control vector consisting of a number of fields that specify the uses and restrictions for that session key. The length of the control vectormay vary.

The control vector is cryptographically coupled with the key at the time of key generation at the KDC.

As a first step, the control vector is passed through a hash function that produces a value whose length is equal to the encryption key length. In essence, a hash function maps values from a larger range into a smaller range with a reasonably uniform spread. Thus, for example, if numbers

in the range 1 to 100 are hashed into numbers in the range 1 to 10, approximately 10% of the source values should map into each of the target values. The hashvalue is then XORed with the master key to produce an output that

is used as the key input for encrypting the session key. Thus,

$$\text{Hash value} = H = h(CV)\text{Key}$$
$$\text{input} = Km \oplus H$$
$$\text{Ciphertext} = E([Km \oplus H], Ks)$$

where is the master key and is the session key. The session key is recovered inplaintext by the reverse operation:

$$D([Km \oplus H], E([Km \oplus H], Ks))$$

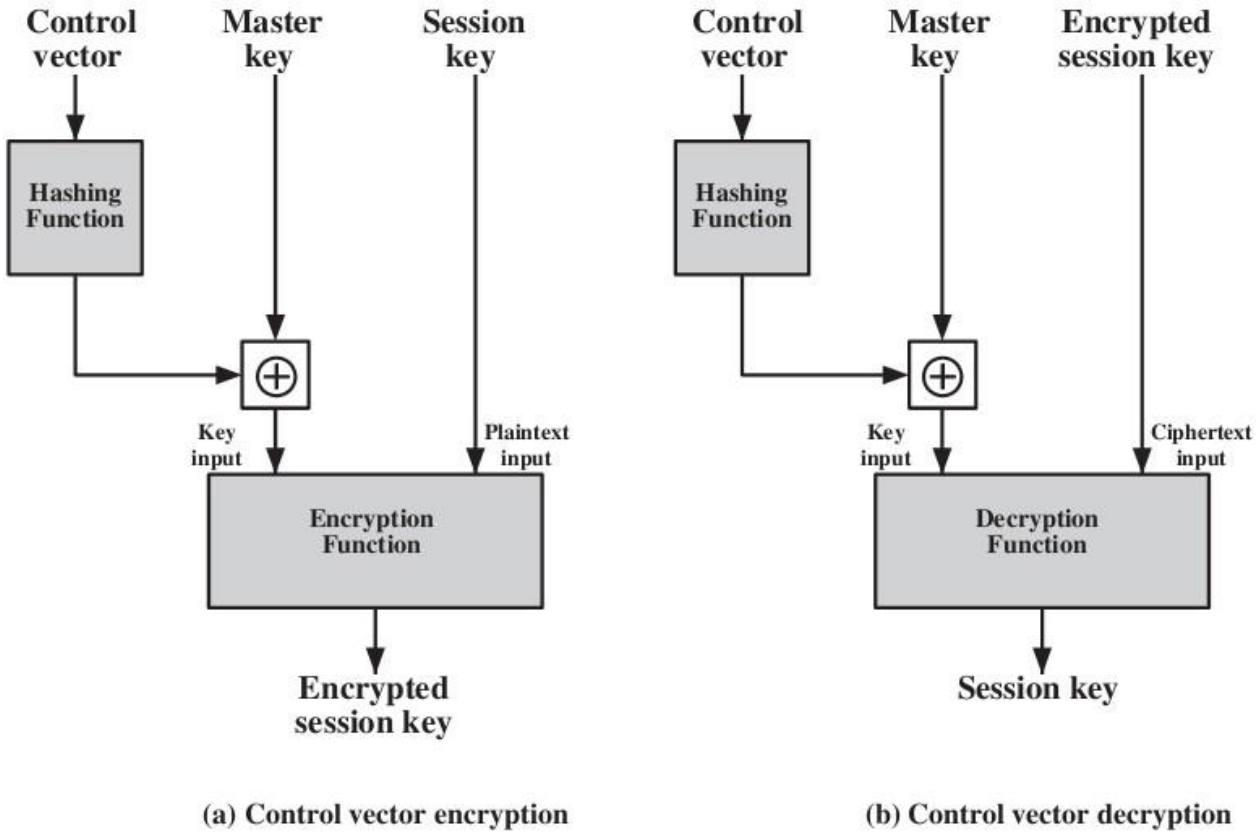


Figure 14.6 Control Vector Encryption and Decryption

When a session key is delivered to a user from the KDC, it is accompanied by the

control vector in clear form. The session key can be recovered only by usingboth the master key that the user shares with the KDC and the control vector. Thus,the linkage between the session key and its control vector is maintained.

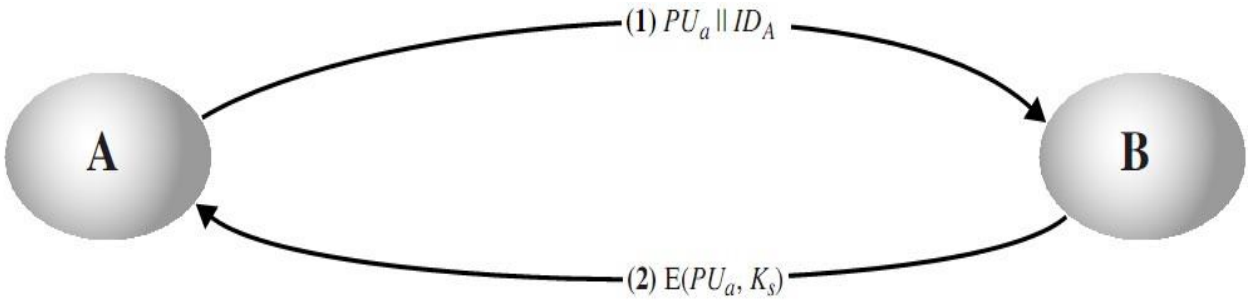
Use of the control vector has two **advantages over use of an 8-bit tag.** First, there is no restriction on length of the control vector, which enables arbitrarily complex controls to be imposed on key use. Second, the control vector is available inclear form at all stages of operation. Thus, control of key use can be exercised in multiple locations.

SYMMETRIC KEY DISTRIBUTION USING ASYMMETRICENCRYPTION

- Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping, tampering, or both, is possible.
- Public-key encryption provides for the distribution of secret keys to be used for conventional encryption.

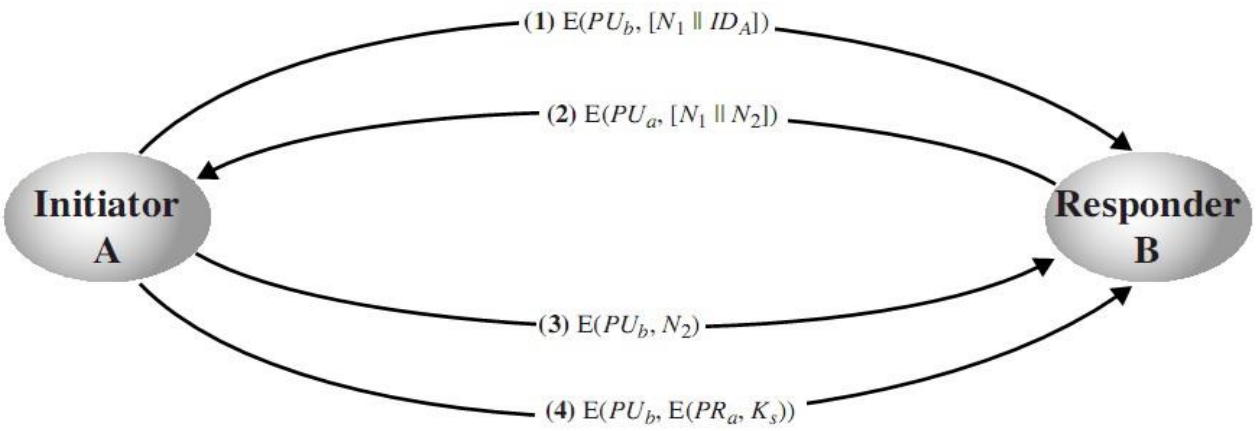
Simple Secret Key Distribution

- A generates a public/private key pair {PUa, PRa} and transmits a message to B consisting of PUa and an identifier of A, IDA
- B generates a secret key, Ks, and transmits it to A, encrypted with A's public key.
- A computes D(PRa, E(PUa, Ks)) to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of Ks.
- A discards PUa and PRa and B discards PUa.



Here third party can intercept messages and then either relay the intercepted messageor substitute another message Such an attack is known as a **man-in-the-middle attack.**

Secret Key Distribution with Confidentiality and Authentication:



- A uses B's public key to encrypt a message to B containing an identifier of A (ID_A) and anonce (N_1), which is used to identify this transaction uniquely
- B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as anew nonce generated by B (N_2) Because only B could have decrypted message (1), thepresence of N_1 in message (2) assures A that the correspondent is B
- A returns N_2 encrypted using B's public key, to assure B that its correspondent is A.
- A selects a secret key K_s and sends M = E(PU_b, E(PR_a, K_s)) to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's privatekey ensures that only A could have sent it.
- B computes D(PU_a, D(PR_b, M)) to recover the secret key.

A Hybrid Scheme:

Yet another way to use public-key encryption to distribute secret keys is a hybrid approach.

- This scheme retains the use of a key distribution center (KDC) that shares a secretmaster key with each user and distributes secret session keys encrypted with the master key.
- A public key scheme is used to distribute the master keys.
- The addition of a public-key layer provides a secure, efficient means of distributingmaster keys.

Distribution of Public Keys:

Several techniques have been proposed for the distribution of public keys, which can mostly be grouped into the categories shown.

- Public announcement
- Publicly available directory
- Public-key authority
- Public-key certificates

Public Announcement of Public Keys

The point of public-key encryption is that the public key is public, hence any participant can send his or her public key to any other participant, or broadcast the key to the community at large. eg. append PGP keys to email messages or post to news groups or email list

Figure 10.1 Uncontrolled Public Key Distribution

Its major weakness is forgery, anyone could pretend to be user A and send a public key to another participant or broadcast such a public key. Until the forgery is discovered they can masquerade as the claimed user.

Publicly Available Directory

- can obtain greater security by registering keys with a public directory
- directory must be trusted with properties:
  - The authority maintains a directory with a {name, public key} entry for each participant.
  - Each participant registers a public key with the directory authority.

- A participant may replace the existing key with a new one at any time because the corresponding private key has been compromised in some way.
- Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

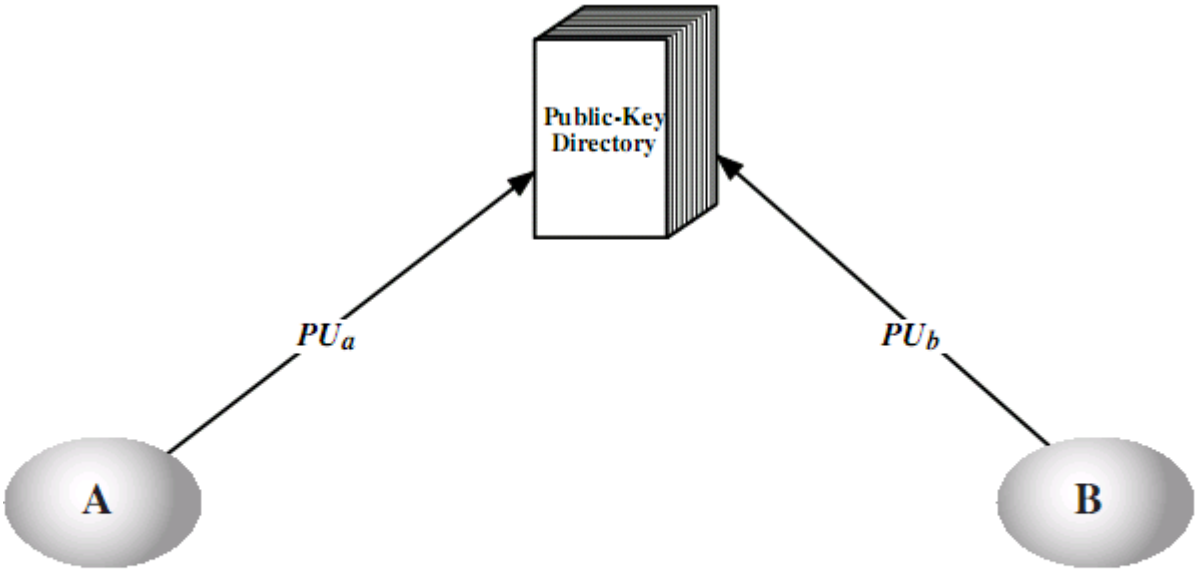


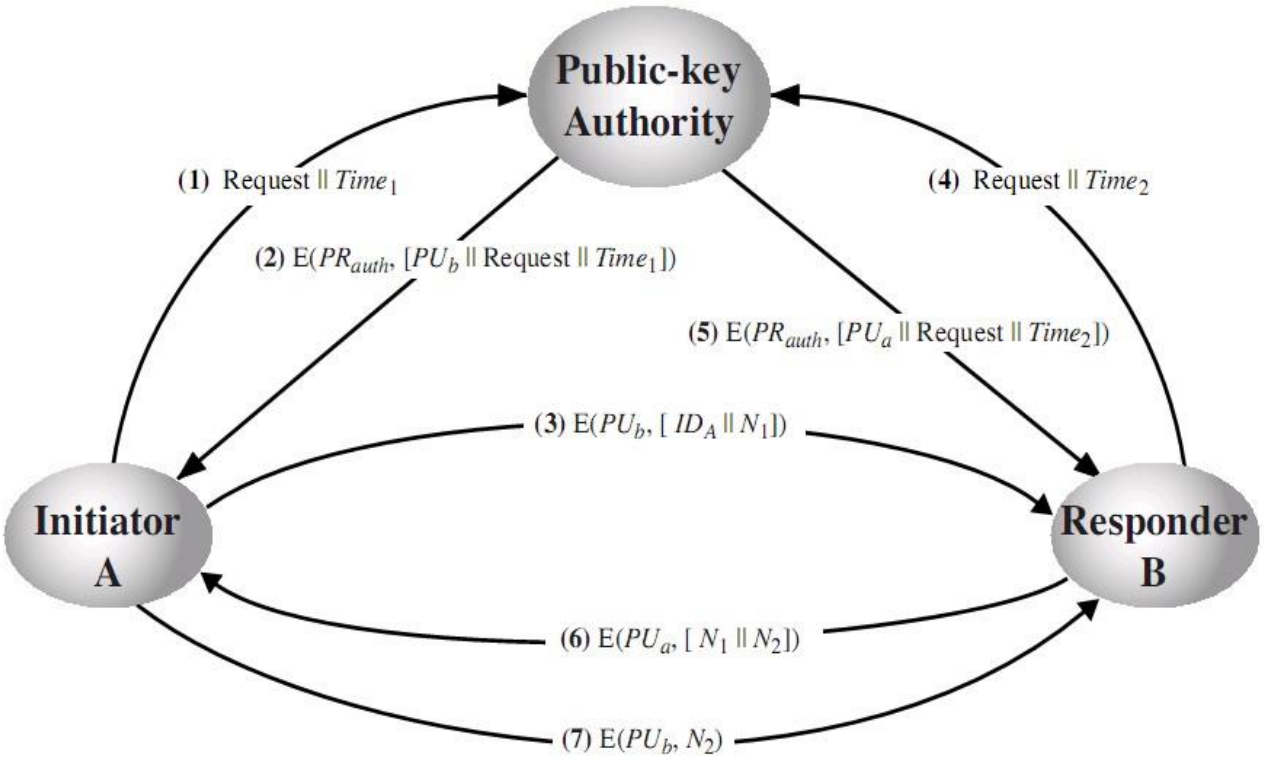
Figure 10.2 Public Key Publication

This scheme is clearly more secure than individual public announcements but still has vulnerabilities.

If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant. Another way to achieve the same end is for the adversary to tamper with the records kept by the authority.

Public-Key Authority:

- Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory.
- It requires users to know the public key for the directory, and that they interact with directory in real-time to obtain any desired public key securely.
- Totally seven messages are required.



1. A sends a timestamped message to the public-key authority containing a request for the current public key of B.

2. The authority responds with a message that is encrypted using the authority's private key, PR_{auth} Thus, A is able to decrypt the message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:

- B's public key, PU_b which A can use to encrypt messages destined for B
- The original request, to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority.
- The original timestamp, so A can determine that this is not an old message from the authority containing a key other than B's current public key.

3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.

4. B retrieves A's public key from the authority in the same manner as A retrieved B's public key.

5. At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

6. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2) Because only B could have decrypted message (3), the presence of N_1 in message (6) assures A that the correspondent is B.

7. A returns N_2, encrypted using B's public key, to assure B that its correspondent is A.

Public-Key Certificates

- A user must appeal to the authority for a public key for every other user that it wishes to contact and it is vulnerable to tampering too.
- Public key certificates can be used to exchange keys without contacting a public-key authority.
- A certificate binds an **identity** to **public key**, with all contents **signed** by a trusted Public-Key or Certificate Authority (CA).
- This can be verified by anyone who knows the public-key authorities public-key.

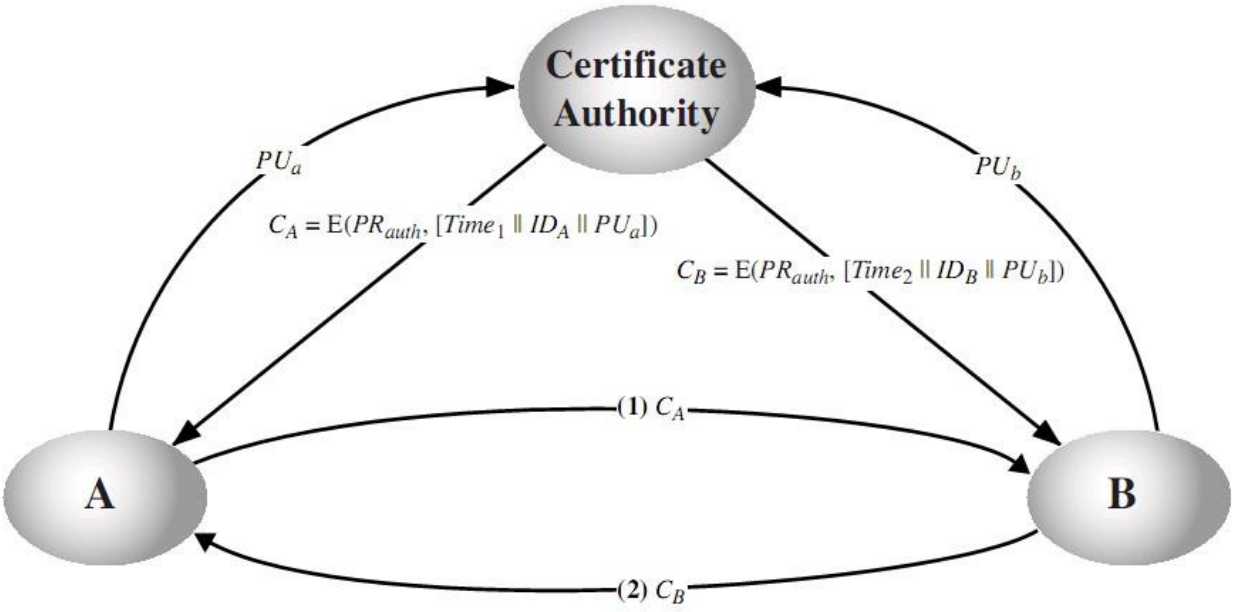
A participant can also convey its key information to another by transmitting its certificate.

Other participants can verify that the certificate was created by the authority. We can place the following requirements on this scheme:

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the currency of the certificate.

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, secure sockets layer (SSL), secure electronic transactions (SET), and S/MIME.



X.509 CERTIFICATES

X.509 is part of the X.500 series of recommendations that define a directory service, being a server or distributed set of servers that maintains a database of information about 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. 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 based on the use of public-key cryptography and digital signatures.

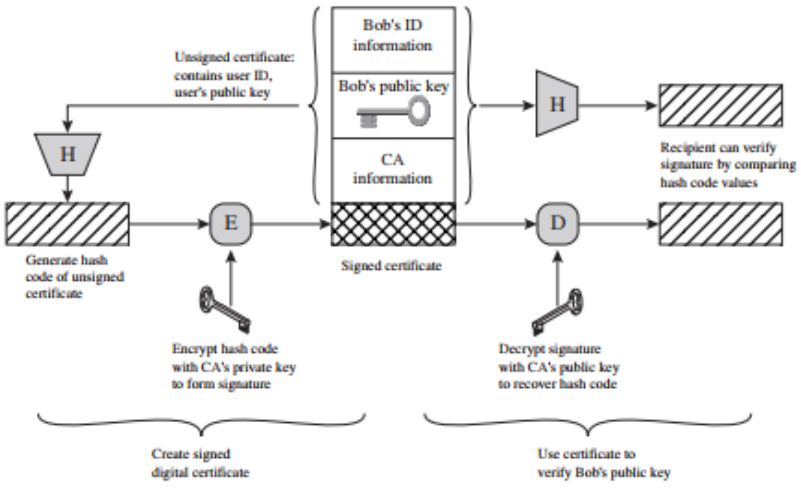


Figure 14.14 Public-Key Certificate Use

The X.509 certificate format is widely used, in for example S/MIME, IP Security and SSL/TLS and SET. X.509 was initially issued in 1988. The standard was subsequently revised

to address some of the security concerns; a revised recommendation was issued in 1993. A third version was issued in 1995 and revised in 2000.

Certificates

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.

The standard uses the notation for a certificate of:

CA<<A>> where the CA signs the certificate for user A with its private key. In more detail CA<<A>> = CA {V, SN, AI, CA, UCA, A, UA, Ap, TA}.

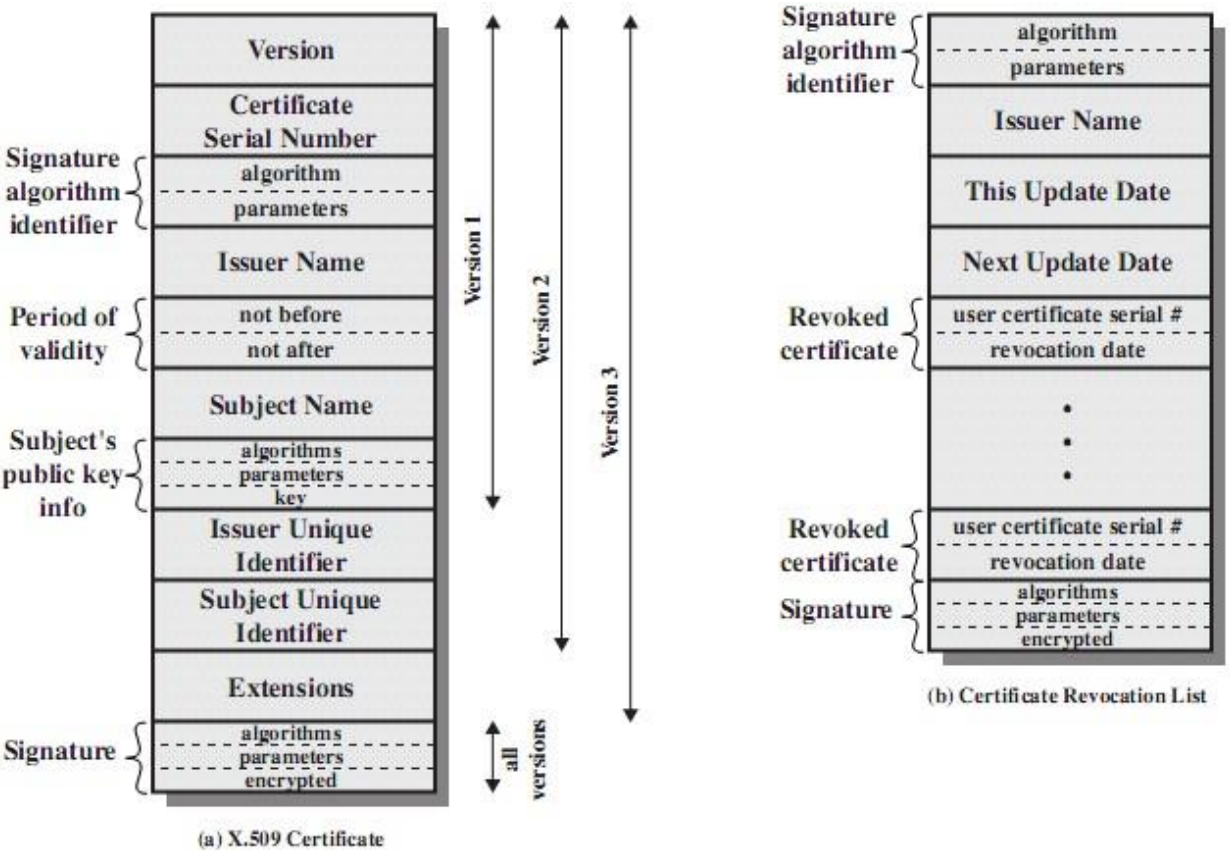


Figure 14.4 X.509 Formats

If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.

- **Version:** Differentiates among successive versions of the certificate format; the default is version 1. If the issuer unique identifier or subject unique identifier are present, the value

must be version 2. 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.
- **Subject unique identifier:** An optional-bit string field used to identify uniquely thesubject i
- **Extensions:** A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.
- **Signature:** Covers all of the other fields of the certificate; it contains the hash code of the other fields encrypted with the CA's private key. This field includes the signature algorithm identifier.

The unique identifier fields were added in version 2 to handle the possible reuse of subject and/or issuer names over time. These fields are rarely used. The standard uses the following notation to define a certificate:

$$CA\langle\langle A\rangle\rangle = CA \{V, SN, AI, CA, UCA, A, UA, Ap, T^A\}$$

where

Y<< X >> = the certificate of user X issued by certification authority Y

Y {I} = the signing of I by Y. It consists of I with an encrypted hash

code appended

V = version of the certificate

SN = serial number of the certificate

AI = identifier of the algorithm used to sign the certificateCA = name of certificate authority

UCA = optional unique identifier of the

CAA = name of user A

UA = optional unique identifier of the user

AAp = public key of user A

T^A = period of validity of the certificate

Obtaining a Certificate

User certificates generated by a CA have the following characteristics:

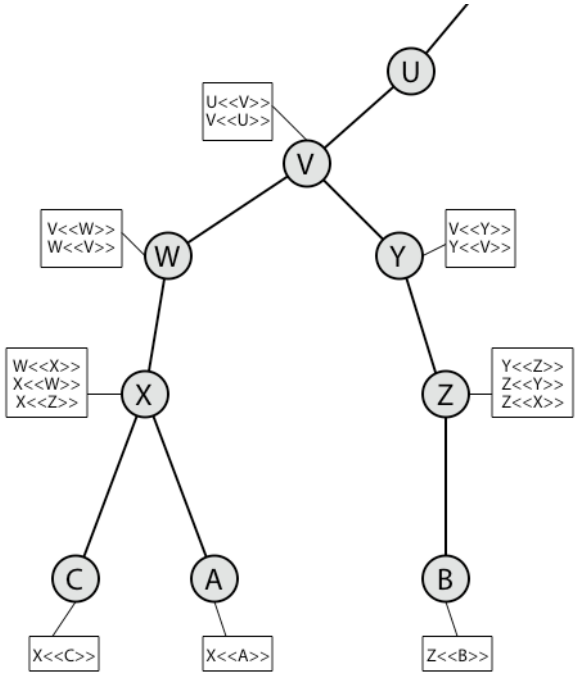
- Any user with access to the public key of the CA can verify the user public key that was certified.
- No party other than the certification authority can modify the certificate with out this being detected.

Because certificates are unforgeable, they can be placed in a directory without the need for the directory to make special efforts to protect them.

If all users subscribe to the same CA, then there is a common trust of that CA. All user certificates can be placed in the directory for access by all users. In addition, a user can transmit his or her certificate directly to other users. In either case, once B is in possession of A's certificate, B has confidence that messages it encrypts with A's public key will be secure from eavesdropping and that messages signed with A's private key are unforgeable.

CA Hierarchy:

If both parties use the same CA, they know its public key and can verify others certificates. If there is a large community of users, it may not be practical for all users to subscribe to the same CA. Hence there has to be some means to form a chain of certifications between the CA's used by the two parties, by the use of client and parent certificates. All these certificates of CAs by CAs need to appear in the directory, and the user needs to know how they are linked to follow a path to another user's public-key certificate. X.509 suggests that CAs be arranged in a hierarchy so that navigation is straightforward. It is assumed that each



client trusts its parent's certificates.

Figure 14.15 illustrates the use of an X.509 hierarchy to mutually verify clients certificates. The connected circles indicate the hierarchical relationship among the CAs; the associated boxes indicate certificates maintained in the directory for each CA entry. The directory entry for each CA includes two types of certificates:

Forward certificates: Certificates of X generated by other CAs,

Reverse certificates: Certificates generated by X that are the certificates of other CAs.In this example, we can track chains of certificates as follows:

A acquires B certificate using chain

X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>

B acquires A certificate using chain:

Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A

>>

Certificate Revocation:

A certificate includes a period of validity. Typically a new certificate is issued just before the expiration of the old one.

In addition, it may be desirable on occasion to revoke a certificate before it expires, for one of a range of following reasons:

1. The user's private key is assumed to be compromised.
2. The user is no longer certified by this CA. Reasons for this include that the subject's name has changed, the certificate is superseded, or the certificate was not issued in conformance with the CA's policies.

3. The CA's certificate is assumed to be compromised.

To support this, each CA must maintain a list consisting of all revoked but not expired certificates issued by that CA, known as the certificate revocation list (CRL). Each certificate revocation list (CRL) posted to the directory is signed by the issuer and includes (as shown in Figure 14.14b previously) the issuer's name, the date the list was created, the date the next CRL is scheduled to be issued, and an entry for each revoked certificate. Each entry consists of the serial number of a certificate and revocation date for that certificate. Because serial numbers are unique within a CA, the serial number is sufficient to identify the certificate.

When a user receives a certificate in a message, the user must determine whether the certificate has been revoked, by checking the directory CRL each time a certificate is received, this often does not happen in practice.

X.509 Version 3

The X.509 version 2 format does not convey all of the information. Rather than continue to add fields to a fixed format, standards developers felt that a more flexible approach was needed. X.509 version 3 includes a number of optional extensions that may be added to the version 2 format. Each extension consists of an extension identifier, a criticality indicator, and an extension value. The criticality indicator indicates whether an extension can be safely ignored or not.

Certificate Extensions

The certificate extensions fall into three main categories:

- **Key and policy information** - convey additional information about the subject and issuer keys, plus indicators of certificate policy. A certificate policy is a named set of rules that indicates the applicability of a certificate to a particular community and/or class of application with common security requirements.
- **Subject and issuer attributes** - support alternative names, in alternative formats, for a certificate subject or certificate issuer and can convey additional information about the certificate subject; eg. postal address, email address, or picture image
- **Certification path constraints** - allow constraint specifications to be included in certificates issued for CA's by other CA's that may restrict the types of certificates that can be issued by the subject CA or that may occur subsequently in a certification chain.

PUBLIC-KEY Infrastructure

RFC 4949 (Internet Security Glossary) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create,

manage, store, distribute, and revoke digital certificates based on asymmetric cryptography. The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys. The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the driving force behind setting up a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet. Figure below shows the interrelationship among the key elements of the PKIX model.

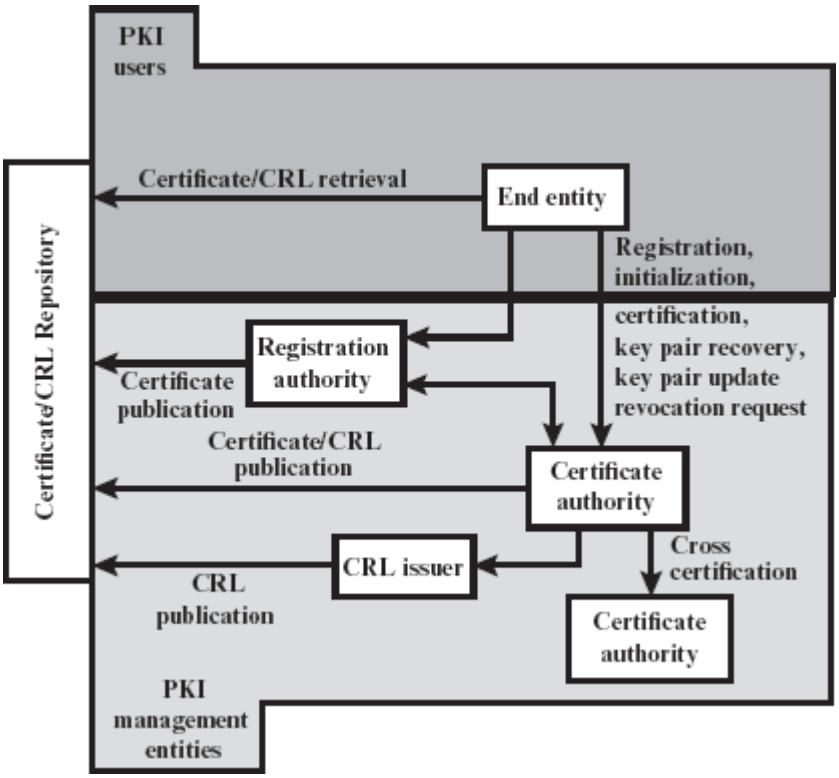


Fig: PKIX Architectural Model

The Elements of PKIX Model are:

- End entity: A generic term used to denote end users, devices (e.g., servers, routers), or any other entity that can be identified in the subject field of a public-key certificate. End entities typically consume and/or support PKI-related services.
- Certification authority (CA): The issuer of certificates and (usually) certificate revocation lists (CRLs). It may also support a variety of administrative functions, although these are oftendelegated to one or more Registration Authorities.
- Registration authority (RA): An optional component that can assume a number of administrative functions from the CA. The RA is often associated with the end entity registration process but canassist in a number of other areas as well.
- CRL issuer: An optional component that a CA can delegate to publish CRLs.
- Repository: A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities.

The PKIX Management Functions are:

- Registration: This is the process whereby a user first makes itself known to a CA (directly or through an RA), prior to that CA issuing a certificate or certificates for that user. Registration begins the process of enrolling in a PKI. Registration usually involves some offline or online procedure for mutual authentication. Typically, the end entity is issued one or more shared secretkeys used for subsequent authentication.
- Initialization: Before a client system can operate securely, it is necessary to install key

materials that have the appropriate relationship with keys stored elsewhere in the infrastructure. For example, the client needs to be securely initialized with the public key and other assured information of the trusted CA(s), to be used in validating certificate paths.

- Certification: This is the process in which a CA issues a certificate for a user's public key, returns that certificate to the user's client system, and/or posts that certificate in a repository.
- Key pair recovery: Key pairs can be used to support digital signature creation and verification, encryption and decryption, or both. When a key pair is used for encryption/decryption, it is important to provide a mechanism to recover the necessary decryption keys when normal accessto the keying material is no longer possible, otherwise it will not be possible to recover the encrypted data. Loss of access to the decryption key can result from forgotten passwords/PINs, corrupted disk drives, damage to hardware tokens, and so on. Key pair recovery allows end entities to restore their encryption/decryption key pair from an authorized key backup facility (typically, the CA that issued the end entity's certificate).
- Key pair update: All key pairs need to be updated regularly (i.e., replaced with a new key pair) and new certificates issued. Update is required when the certificate lifetime expires and as a result ofcertificate revocation.
- Revocation request: An authorized person advises a CA of an abnormal situation requiring certificate revocation. Reasons for revocation include private key compromise, change in affiliation, and name change.
- Cross certification: Two CAs exchange information used in establishing a cross-certificate. A cross- certificate is a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.

KERBEROS

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Kerberos relies exclusively on conventional encryption, making no use of public-key encryption.

Motivation

A distributed architecture consists of dedicated user workstations (clients) and distributed or centralized servers. In this environment, there are three approaches to security:

- Rely on each individual client workstation to assure the identity of its user or users andrely on each server to enforce a security policy based on user identification (ID).
- Require that client systems authenticate themselves to servers, but trust the clientsystem concerning the identity of its user.
- Require the user to prove his or her identity for each service invoked. Also require thatservers prove their identity to clients.

The following are the **requirements for Kerberos**:

- **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- **Reliable:** For all services that rely on Kerberos for access control, lack of

availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ distributed server architecture,with one system able to back up another.

- **Transparent:** Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.
- **Scalable:** The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

To support these requirements, the overall scheme of Kerberos is that of a trusted third-party authentication service that uses a protocol based on Needham and Schroeder.It is trusted in the sense that clients and servers trust Kerberos to mediate their mutual authentication. Assuming the Kerberos protocol is well designed, and then the authentication service is secure if the Kerberos server itself is secure.

Two versions of Kerberos are in common use. **Version 4** and **Version**

5<u>Kerberos Version 4</u>

Version 4 of Kerberos makes use of DES, in a rather elaborate protocol, to provide the authentication service

1.A Simple Authentication Dialogue

In an unprotected network environment, any client can apply to any server for service. The obvious security risk is that of impersonation. To counter this threat, servers must be able to confirm the identities of clients who request service. But in an open environment, this placesa substantial burden on each server.

An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database. In addition, the AS shares a unique secret key with each server. The simple authentication dialogue is as follows:

1. C >> AS: $ID_c||P_c||ID_v$

2. AS >> C: Ticket

3. C >> V: $ID_c||$Ticket

Ticket=

$EK_v(ID_c||ADc||ID_v)$

C : Client,
AS : Authentication Server,
V : Server, ID_c : ID of the client,P_c : Password of the client,
AD_c : Address of client, ID_v : ID of the server,K_v : secret key shared by AS and V,
|| : concatenation.

2.A More Secure Authentication Dialogue

There are two major problems associated with the previous approach:

- Plaintext transmission of the password.

- Each time a user has to enter the password.

To solve these problems, we introduce a scheme for avoiding plaintext passwords, anda new server, known as ticket granting server (TGS). The hypothetical scenario is as follows:

Once per user logon session:-

1. C >> AS: $ID_c||ID_{tgs}$
2. AS >> C: $Ek_c(Ticket_{tgs})$

Once per type of service:

3. C >> TGS: $ID_c||ID_v||Ticket_{tgs}$
4. TGS >> C: $ticket_v$

Once per service session:

5. C >> V: $ID_c||\ Ticket_v$

$Ticket_{tgs}$=
$Ekt_{gs}(ID_c||AD_c||IDt_{gs}||TS_1||Lifetime_1)$
$Ticket_v$=
$Ek_v(ID_c||AD_c||ID_v||TS2||Lifetime_2)$

C: Client, AS: Authentication Server, V: Server,
IDc : ID of the client, Pc:Password of the client, ADc: Address of client,IDv : ID of the server, Kv: secret key shared by AS and V,
|| : concatenation, IDtgs: ID of the TGS server, TS1, TS2: time stamps, lifetime:lifetime of the ticket.

The new service, TGS, issues tickets to users who have been authenticated to AS. Thus, the user first requests a ticket-granting ticket ($Ticket_{tgs}$) from the AS. The client module in the user workstation saves this ticket.

Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself. The TGS then grants a ticket for the particular service. The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested.

Let us look at the details of this scheme:

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID and password to the AS, together with the TGS ID, indicating a request to use the TGS service
2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password.

When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message.

If the correct password is supplied, the ticket is successfully recovered.

Because only the correct user should know the password, only the correct user can recover the ticket. Thus, we have used the password to obtain credentials from Kerberos

without having to transmit the password in plaintext. Now that the client has a ticket-granting ticket, access to any server can be obtained with steps 3 and 4:

3. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket
4. The TGS decrypts the incoming ticket and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.

The service-granting ticket has the same structure as the ticket-granting ticket. Indeed, because the TGS is a server, we would expect that the same elements are needed to authenticate a client to the TGS and to authenticate a client to an application server.

Again, the ticket contains a timestamp and lifetime. If the user wants access to the same service at a later time, the client can simply use the previously acquired service-granting ticket and need not bother the user for a password.

Note that the ticket is encrypted with a secret key (K_v) known only to the TGS and the server, preventing alteration.

Finally, with a particular service-granting ticket, the client can gain access to the corresponding service with step 5:

5. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service-granting ticket. The server authenticates by using the contents of the ticket.

This new scenario satisfies the two requirements of only one password query per user session and protection of the user password.

Kerberos V4 Authentication Dialogue Message Exchange

Two additional problems remain in the more secure authentication dialogue:

- Lifetime associated with the ticket granting ticket. If the lifetime is very short, then the user will be repeatedly asked for a password. If the lifetime is long, then the opponent has the greater opportunity for replay.

- Requirement for the servers to authenticate themselves to users.

The actual Kerberos protocol version 4 is as follows
:

- A basic third-party authentication scheme
- Have an Authentication Server (AS)
    - Users initially negotiate with AS to identify self
    - AS provides a non-corruptible authentication credential (ticket grantingticket TGT)

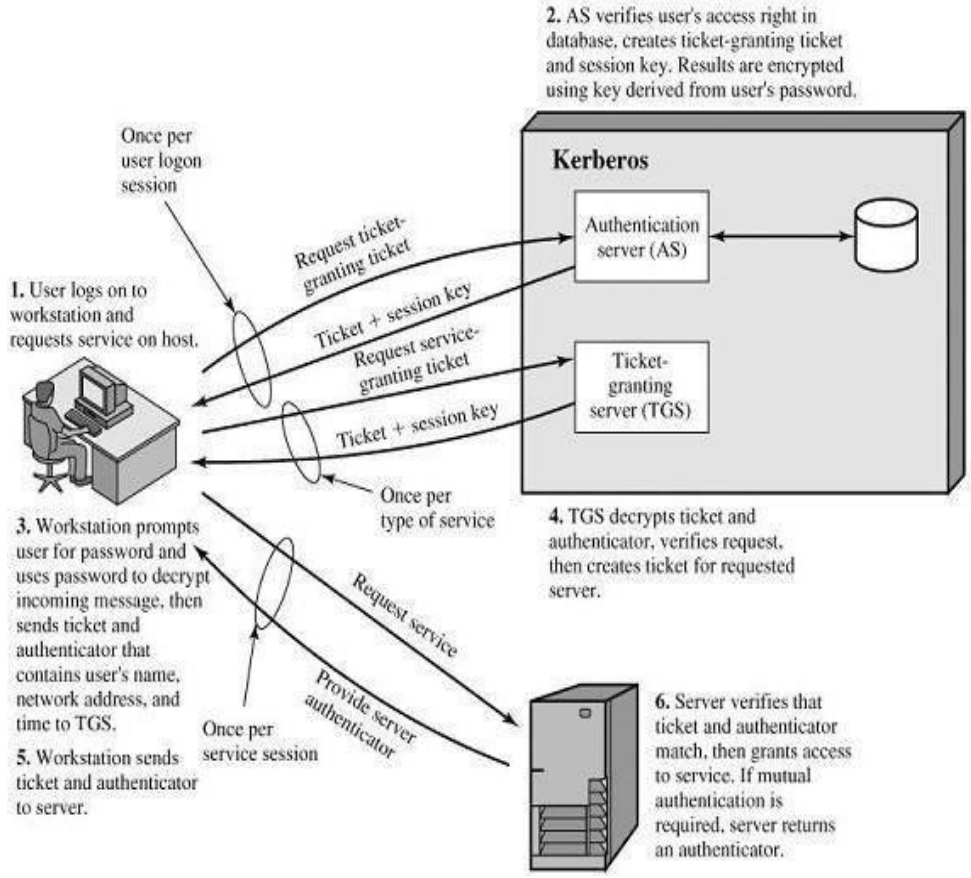
- Have a Ticket Granting
    - Users subsequently request access to other services from TGS on basis of users TGT

(a) Authentication service exchange: to obtain ticket granting ticket
(1) $C \rightarrow AS : ID_C \text{ II } ID_{tgs} \text{ II } TS_1$ (2) $AS \rightarrow C : EKc [K_{c,tgs} \text{ II } ID_{tgs} \text{ II } TS_2 \text{ II Lifetime}_2 \text{ II Ticket}_{tgs}]$
(b) Ticket-Granting Service Exchange: to obtain service-granting ticket

(3) $C \rightarrow TGS: ID_v \text{ II Ticket}_{tgs} \text{ II Authenticator}_c$ (4) $TGS \rightarrow C: EK_{c,tgs}[K_{c,y} \text{ II } ID_v \text{ II } TS_4 \text{ II Ticket}_v]$ $\text{Ticket}_{tgs} = E_{K,tgs}[K_{c,tgs} \text{ II } ID_C \text{ II } AD_C \text{ II } ID_{tgs} \text{ IITS}_2 \text{ II Lifetime}_2]$ $\text{Ticket}_v = E_{Kv}[K_{c,v} \text{ II } ID_C \text{ II } AD_C \text{ II } ID_v \text{ IITS}_4 \text{ II Lifetime}_4]$ $\text{Authenticator}_C = E_{Ktgs} [ID_C \text{ II } AD_C \text{ II } TS_3]$
(c) Client/Server Authentication Exchange: to obtain service
(5) $C \rightarrow V : \text{Ticket}_v \text{ II Authenticator}_c$ (6) $V \rightarrow C: E_{kc,v}[TS_5 +1]$ $\text{Ticket}_v = EK_v[K_{c,v} \text{ II } ID_C \text{ II } AD_C \text{ II } Id_v \text{ II } TS_4 \text{ II Lifetime}_4]$ $\text{Authenticator}_c = EK_{tgs} [ID_C \text{ II } AD_C \text{ II } TS_3]$

Kerberos 4 Overview

Fig 4.1 Overview of Kerberos 4



Kerberos Realms and Multiple Kerberi

A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:

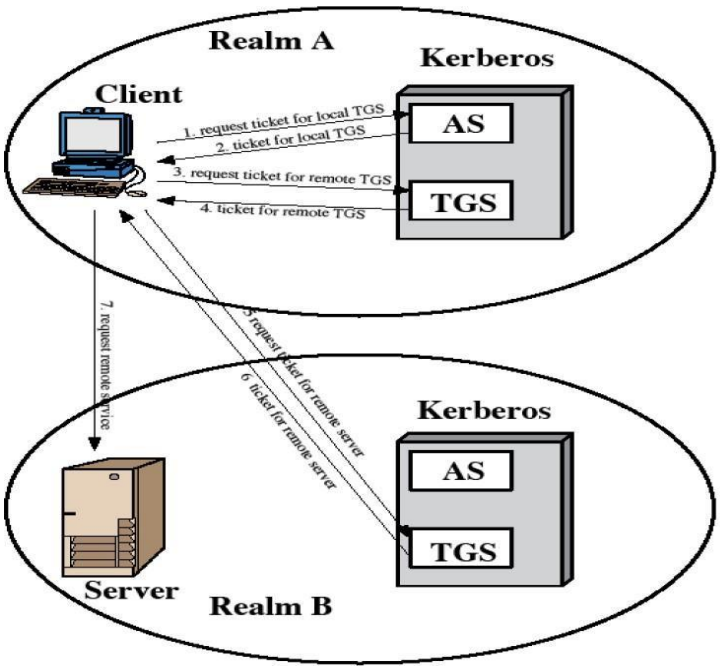
4. The Kerberos server must have the user ID and hashed passwords of all participatingusers in its database. All users are registered with the Kerberos server.
5. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.

Such an environment is referred to as a **Kerberos**

realmThe concept of *realm* can be explained as

follows.

Fig .Request for service in another Realm



A Kerberos realm is a set of managed nodes that share the same Kerberos database. The Kerberos database resides on the Kerberos master computer system, which should be keptin a physically secure room. A read-only copy of the Kerberos database might also reside on other Kerberos computer systems.

However, all changes to the database must be made on the master computer system. Changing or accessing the contents of a Kerberos database requires the Kerberos master

password. A related concept is that of a Kerberos principal, which is a service or user that is known to the Kerberos system.

Each Kerberos principal is identified by its principal name. Principal names consist of three parts: a service or user name, an instance name, and a realm name. Networks of clients and servers under different administrative organizations typically constitute different realms.

That is, it generally is not practical, or does not conform to administrative policy, to have users and servers in one administrative domain registered with a Kerberos server elsewhere.

However, users in one realm may need access to servers in other realms, and some servers may be willing to provide service to users from other realms, provided that those users are authenticated.

Kerberos provides a mechanism for supporting such inter realm authentication. For two realms to support inter realm authentication, a third requirement is added:

6. The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.

The scheme requires that the Kerberos server in one realm trust the Kerberos server in the other realm to authenticate its users. Furthermore, the participating servers in the second realm must also be willing to trust the Kerberos server in the first realm.

The details of the exchanges illustrated in Fig 2are as follows:

$C \rightarrow AS$:ID_C II ID_{tgs} II TS_1

$AS \rightarrow C$:$EK_c[K_{c,tgs}$ ii ID_{tgs} II TS_2 II $Lifetime_2$ II

$Ticket_{tgs}C \rightarrow TGS$:ID_{tgsrem} II $Ticket_{tgs}$ II $Authenticator_c$

$TGS \rightarrow C$:E Kc,tgs[Kc,tgsrem II IDtgsrem II TS4

II $Ticket_{tgsrem}C \rightarrow TGS_{rem}$:ID_{vrem} II $Ticket_{tgsrem}$

II $Authenticator_c$

$TGS_{rem} \rightarrow C$:$EK_{c,tgsrem}$ [$K_{c,vrem}$ II ID_{vrem} II TS_6 II $Ticket_{vrem}$:

$C \rightarrow V_{rem}$:$Ticket_{vrem}$ II$Authenticator_c$

Differences between Versions 4 and 5

Version 5 is intended to address the limitations of version 4 in two areas: environmental shortcomings and technical deficiencies.

Environmental shortcomings:

7. **Encryption system dependence:**

Version 4 requires the use of DES. In version 5, ciphertext is tagged with an encryption type identifier so that any encryption technique may be used.

8. Internet protocol dependence:

Version 4 requires the use of Internet Protocol (IP) addresses. Version 5 network addresses are tagged with type and length, allowing any network address type to be used.

9. Message byte ordering:

In version 4, the sender of a message employs a byte ordering of its own choosing and tags the message to indicate least significant byte in lowest address In version 5, all message structures are defined using Abstract Syntax Notation One (ASN.1) and Basic Encoding Rules (BER), which provide an unambiguous byte ordering.

10. Ticket lifetime:

Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. In version 5, tickets include an explicit start time and end time, allowing tickets with arbitrary lifetimes.

11. Authentication forwarding:

Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. Version 5 provides this capability.

Technical deficiencies in the version 4 protocol:

- Double encryption
- PCBC encryption
- Session keys
- Password attacks

The Version 5 Authentication Dialogue

(a) Authentication Service Exchange: to obtain ticket-granting ticket

(1) $C \rightarrow AS$: Options II ID_c II $Realm_c$ II Times II $Nonce_1$

(2) $AS \rightarrow C$: $Realm_c$ II ID_c II $Ticket_{tgs}$ II EK_c [$K_{c,tgs}$ II Times II $Nonce_1$ II $Realm_{tgs}$ II ID_{tgs}]$Ticket_{tgs} = EK_{tgs}$ [Flags II $K_{c,tgs}$ II $Realm_c$ II ID_c II AD_c II Times]

(b) Ticket – Granting Service Exchange:to obtain service-granting ticket

(3) $C \rightarrow TGS$: Optionns II ID_v II Times II $Nonce_1$

(4) $TGS \rightarrow C$: $Realm_c$ II ID_c II $Ticket_v$ II $EK_{c,tgs}$[$K_{c,v}$ II Times II $Nonce_2$ II $Realm_v$ II ID_v]$Ticket_{tgs} = EK_{tgs}$[Flags II $K_{c,tgs}$ II $Realm_c$ II ID_c II AD_c II Times]

$Ticket_v = Ek_v$[[Flags II $K_{c,v}$ II $Realm_c$ II ID_c II AD_c II Times]

$Authenticator_c = EK_{c,tgs}$[$ID_c$ II $Realm_c$ II TS_1]

(c) Client/Server AUTHENTICATION Exchange: to obtain service

(5) $C \rightarrow V$: Options II $Ticket_v$ II $Authenticator_c$

(6) $V \rightarrow C$: $EK_{c,v}$ [TS_2 II subkey II Seq #]

Ticketv = EK_v[Flags II $K_{c,v}$ II $Realm_c$ II ID_c II AD_c II Times]

$Authenticator_c = E_{Kc,v}$[$ID_c$ II $Realm_c$ II TS_2 II Subkey II Seq#]

First, consider the authentication service exchange. Message (1) is a client request for a ticket-granting ticket. It includes the ID of the user and the TGS.

The following new elements are added:

- Realm: Indicates realm of user
- Options: Used to request that certain flags be set in the returned ticket
- Times: Used by the client to request the following time settings in the ticket:
    - from : the desired start time for the requested ticket
    - till : the requested expiration time for the requested ticket
    - r_{time} : requested renew-till time

Nonce: A random value to be repeated in message (2) to assure that the response is fresh and has not been replaced by an opponent .

Message (2) returns a ticket-granting ticket, identifying information for the client, and a block encrypted using the encryption key based on the user's password. This block includes the session key to be used between the client and the TGS, times specified in message (1), the nonce from message (1), and TGS identifying information.

The ticket itself includes the session key, identifying information for the client, the requested time values, and flags that reflect the status of this ticket and the requested options.

Let us now compare the ticket-granting service exchange for versions 4 and 5.

We see that message (3) for both versions include an authenticator, a ticket, and the name of the requested service.

In addition, version 5 includes requested times and options for the ticket and a nonce, all with functions similar to those of message (1). The authenticator itself is essentially the same as the one used in version 4.

The authenticator itself is essentially the same as the one used in version 4.

Message (4) has the same structure as message (2), returning a ticket plus information needed by the client, the latter encrypted with the session key now shared by the client and the TGS.

Finally, for the client/server authentication exchange, several new features appear in version 5. In message (5), the client may request as an option that mutual authentication is required. The authenticator includes several new fields as follows:

- **Subkey**: The client's choice for an encryption key to be used to protect this specific application session. If this field is omitted, the session key from the ticket (Kc,v) is used.

- **Sequence number**: An optional field that specifies the starting sequence number to be used by the server for messages sent to the client during this session. Messages maybe sequence numbered to detect replays.

CRYPTOGRAPHY AND NETWORK SECURITY
UNIT 4

WEB SECURITY

Usage of internet for transferring or retrieving the data has got many benefits like speed, reliability, security etc. Much of the Internet's success and popularity lies in the fact that it is an open global network. At the same time, the fact that it is open and global makes it not very secure. The unique nature of the Internet makes exchanging information and transacting business over it inherently dangerous.

For the exchange of information and for commerce to be secure on any network, especially the Internet, a system or process must be put in place that satisfies requirements for confidentiality, access control, authentication, integrity, and non-repudiation. These requirements are achieved on the Web through the use of encryption and by employing digital signature technology. There are many examples on the Web of the practical application of encryption. One of the most important is the SSL protocol.

A summary of types of security threats faced in using the Web is given below:

Web Security Threats:

Table 16.1 provides a summary of the types of security threats faced when using the Web. One way to group these threats is in terms of passive and active attacks. Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted. Active attacks include impersonating another user, altering messages in transit between client and server, and altering information on a Web site. Another way to classify Web security threats is in terms of the location of the threat: Web server, Web browser, and network traffic between browser and server.

Table 16.1 A Comparison of Threats on the Web

	Threats	Consequences	Countermeasures
Integrity	• Modification of user data • Trojan horse browser • Modification of memory • Modification of message traffic in transit	• Loss of information • Compromise of machine • Vulnerabilty to all other threats	Cryptographic checksums
Confidentiality	• Eavesdropping on the net • Theft of info from server • Theft of data from client • Info about network configuration • Info about which client talks to server	• Loss of information • Loss of privacy	Encryption, Web proxies
Denial of Service	• Killing of user threads • Flooding machine with bogus requests • Filling up disk or memory • Isolating machine by DNS attacks	• Disruptive • Annoying • Prevent user from getting work done	Difficult to prevent
Authentication	• Impersonation of legitimate users • Data forgery	• Misrepresentation of user • Belief that false information is valid	Cryptographic techniques

SECURE SOCKET LAYER

- 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.
- SSL/TLS includes protocol mechanisms to enable two TCP users to determine the security mechanisms and services they will use.
- Netscape originated SSL.

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 in Figure 16.2.

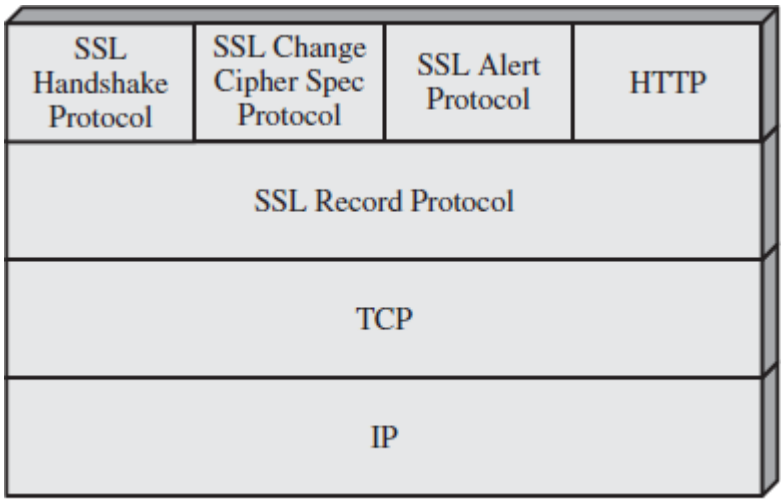


Figure 16.2 SSL Protocol Stack

The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer 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.

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.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..

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 data sent 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.

- **Sequence numbers:** Each party maintains separate sequence numbers for transmitted and received messages for each connection.

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 16.3 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 before being delivered to higher-level users.

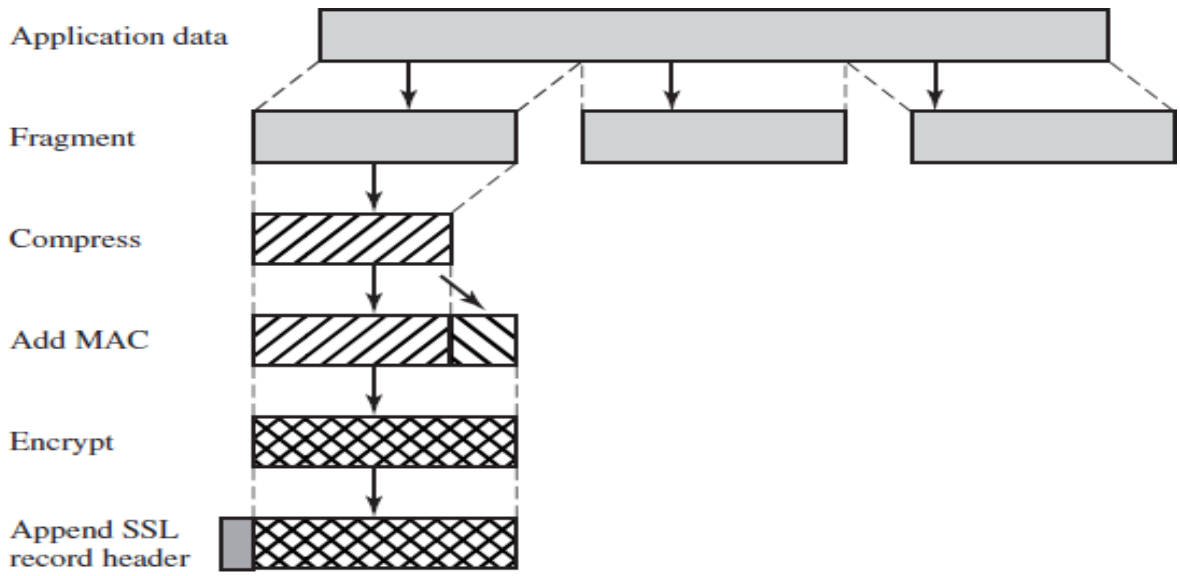


Figure 16.3 SSL Record Protocol Operation

- 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 next step is perform encryption and adds a header

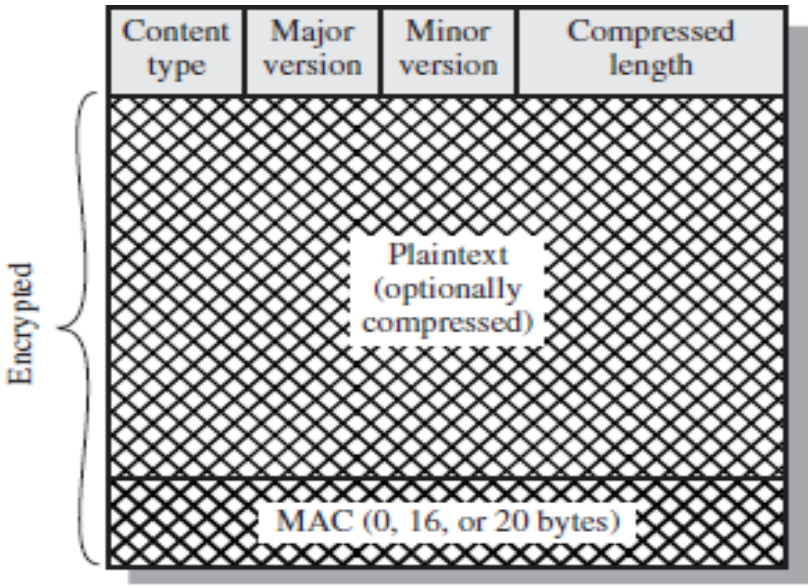


Figure 16.4 SSL Record Format

SSL Handshake Protocol

This phase is used to initiate a logical connection between client and server

The Handshake Protocol consists of a series of messages exchanged by client and server. All of these have the format shown in Figure 16.5c. Each message has three fields:

- **Type (1 byte):** Indicates one of 10 messages. Table 16.2 lists the defined message types.
- **Length (3 bytes):** The length of the message in bytes.
- **Content (≥ 0 bytes):** The parameters associated with this message; these are listed in Table 16.2.

Table 16.2 SSL Handshake Protocol Message Types

Message Type	Parameters
`hello_request`	null
`client_hello`	version, random, session id, cipher suite, compression method
`server_hello`	version, random, session id, cipher suite, compression method
`certificate`	chain of X.509v3 certificates
`server_key_exchange`	parameters, signature
`certificate_request`	type, authorities
`server_done`	null
`certificate_verify`	signature
`client_key_exchange`	parameters, signature
`finished`	hash value

- It consists of 4 phases
    1. Establish Security Capabilities
    2. Server Authentication and Key Exchange
    3. Client Authentication and Key Exchange
    4. Finish

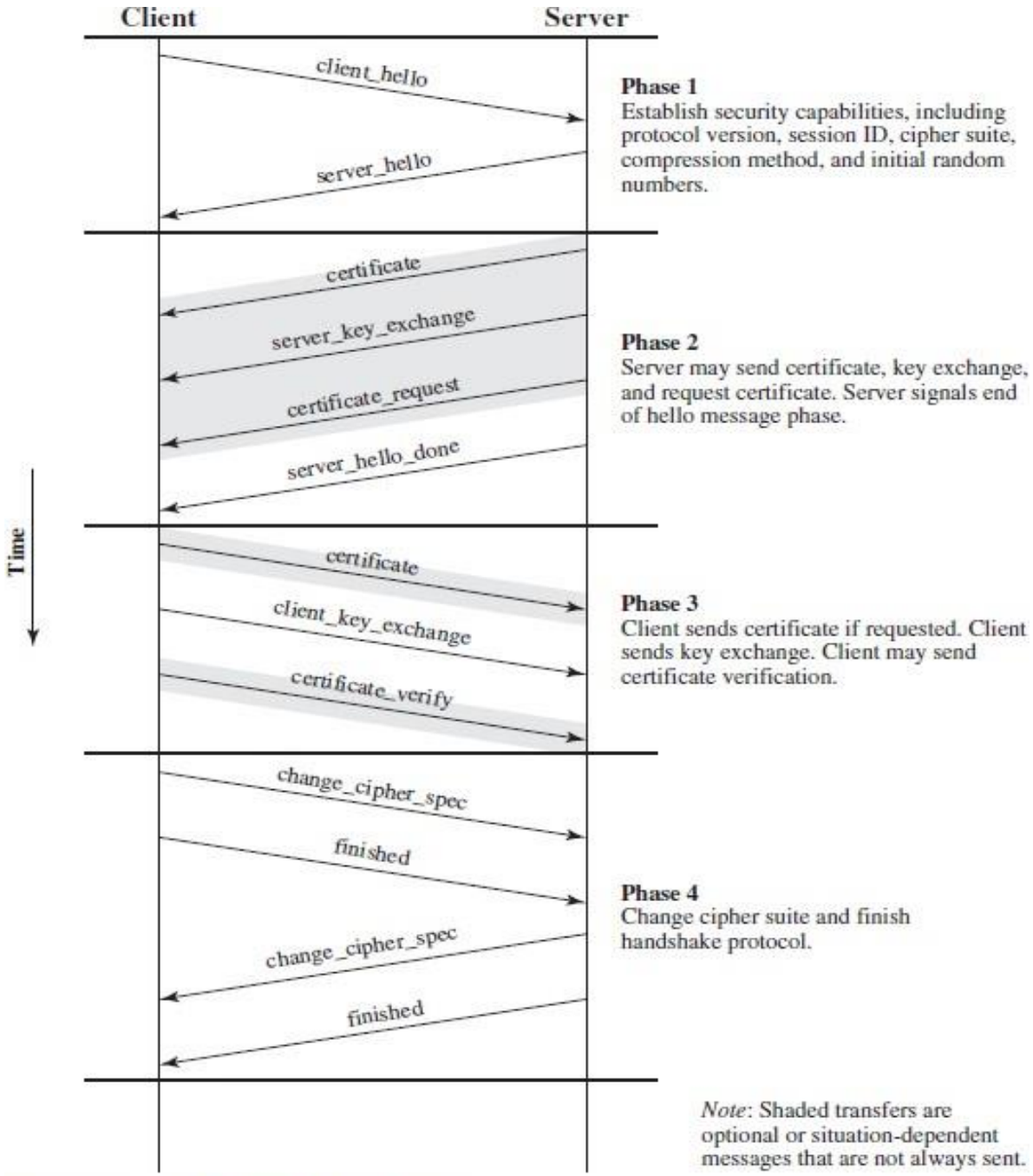


Figure 16.6 Handshake Protocol Action

The exchange is initiated by the client, which sends a **client_hello message** with the followingparameters:

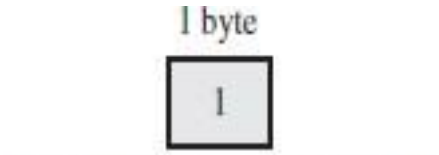
- **Version:** The highest SSL version understood by the client.
- **Random:** A client-generated random structure consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator.

These values serve as nonces and are used during key exchange to prevent replay attacks.

- **Session ID:** A variable-length session identifier. A nonzero value indicates that the client wishes to update the parameters of an existing connection or to create a new connection on this session. A zero value indicates that the client wishes to establish a new connection on a new session.
- **Cipher Suite:** This is a list that contains the combinations of cryptographic algorithms supported by the client, in decreasing order of preference. Each element of the list (each cipher suite) defines both a key exchange algorithm and a Cipher Spec; these are discussed subsequently.
- **Compression Method**: This is a list of the compression methods the client supports.

Change Cipher Spec Protocol

- The Change Cipher Spec Protocol is one of the three SSL-specific protocols that use theSSL Record Protocol, and it is the simplest.
- This protocol consists of a single message (Figure 16.5a), which consists of a single bytewith the value 1.
- The sole purpose of this message is to cause the pending state to be copied into thecurrent state,

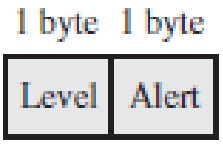


(a) Change Cipher Spec Protocol

Alert Protocol

- The Alert Protocol is used to convey SSL-related alerts to the peer entity
- Each message in this protocol consists of two bytes (Figure 16.5b). The first byte takesthe value warning (1) or fatal (2) to convey the severity of the message.

- If the level is fatal, SSL immediately terminates the connection
- The second byte contains a code that indicates the specific alert.



(b) Alert Protocol

Example Alerts

fatal: unexpected message, bad record mac, decompression failure, handshake failure, illegalparameter

warning: close notify, no certificate, bad certificate, unsupported certificate, certificate revoked,certificate expired, certificate unknown

unexpected_message: An inappropriate message was received.

bad_record_mac: An incorrect MAC was received.

TRANSPORT LAYER SECURITY

TLS was released in response to the Internet community's demands for a standardized protocol. TLS (Transport Layer Security), defined in RFC 2246, is a protocol for establishing a secure connection between a client and a server.

TLS (Transport Layer Security) is capable of authenticating both the client and the server and creating a encrypted connection between the two.

Many protocols use TLS (Transport Layer Security) to establish secure connections, including HTTP, IMAP, POP3, and SMTP.

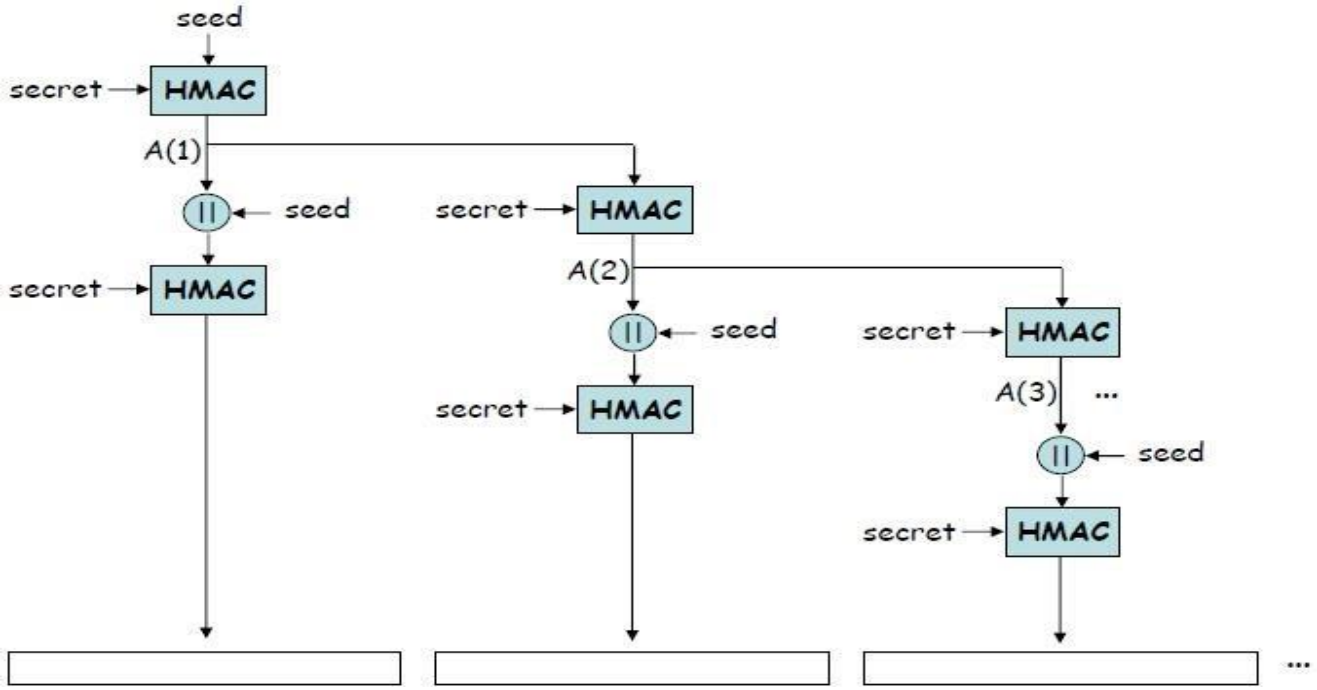
The TLS Handshake Protocol first negotiates key exchange using an asymmetric algorithm suchas RSA or Diffie- Hellman .

TLS is very similar to SSLv3.

There are some minor differences ranging from protocol version numbers to generation of keymaterial.

Version Number: The TLS Record Format is the same as that of the SSL Record Format and the fields in the header have the same meanings. The one difference is in version values. For the current version of TLS, the Major Version is 3 and the Minor Version is 3.

Message Authentication Code: Two differences arise one being the actual algorithm and the other being scope of MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104. SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. For TLS, the MAC calculation encompasses the fields.




```
HMAC hash(MAC write secret, seq num || TLSCompressed.type ||
          TLSCompressed.version || TLSCompressed.length ||
          TLSCompressed.fragment)
```

The MAC calculation covers all of the fields covered by the SSLv3 calculation, plus thefield TLSCompressed. version, which is the version of the protocol being employed.

Pseudorandom Function: TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation. The PRF is based on the following data expansion function:

```
P hash(secret, seed) = HMAC hash(secret, A(1)  || seed) ||
                       HMAC hash(secret, A(2)  || seed) ||
                       HMAC hash(secret, A(3) || seed) || ...
```

where A() is defined as

```
A(0) = seed
A(i) = HMAC_hash (secret, A(i - 1))
```

Alert Codes

TLS supports all of the alert codes defined in SSLv3 with the exception of no_certificate .A number of additional codes is defined in TLS. Some of them are

- **record_overflow**
- **unknown_ca**
- **access_denied**
- **protocol_version**
- **internal_error**
- **decrypt_error**

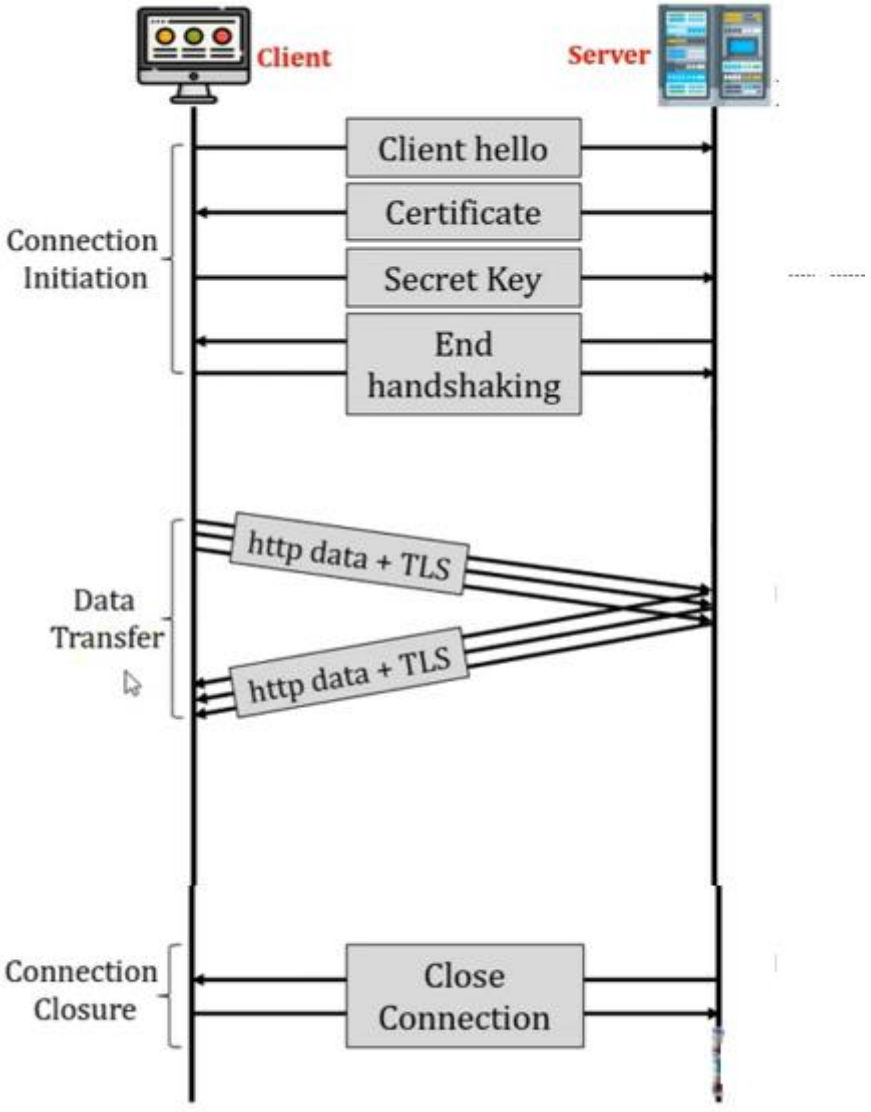
HTTPS

HTTPS (HTTP over SSL) refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server. The HTTPS capability is built into all modern Web browsers. Its use depends on the Web server supporting HTTPS communication. For example, some search engines do not support HTTPS. Google provides HTTPS as an option:
https://google.com.
The principal difference seen by a user of a Web browser is that URL (uniform resource locator) addresses begin with https:// rather than http://. A normal HTTP connection uses port 80. If HTTPS is specified, port 443 is used, which invokes SSL.
When HTTPS is used, the following elements of the communication are encrypted:
• URL of the requested document
• Contents of the document
• Contents of browser forms (filled in by browser user)
• Cookies sent from browser to server and from server to browser
• Contents of HTTP header
HTTPS is documented in RFC 2818, *HTTP Over TLS*. There is no fundamental change in using HTTP over either SSL or TLS, and both implementations are
referred to as HTTPS.

Connection Initiation

For HTTPS, the agent acting as the HTTP client also acts as the TLS client. The client initiates a connection to the server on the appropriate port and then sends the TLS ClientHello to begin the TLS handshake. When the TLS handshake has finished, the client may then initiate the first HTTP request. All HTTP data is to be sent as TLS application data. Normal HTTP behavior, including retained connections, should be followed.
There are three levels of awareness of a connection in HTTPS. At the HTTP level, an HTTP

client requests a connection to an HTTP server by sending a connection request to the next lowest layer. Typically, the next lowest layer is TCP, but it also may be TLS/SSL. At the level of TLS, a session is established between a TLS client and a TLS server. This session can support one or more connections at any time. As we have seen, a TLS request to establish a connection begins with the establishment of a TCP connection between the TCP entity on the client side and the TCP entity on the server side.



Connection Closure

An HTTP client or server can indicate the closing of a connection by including the following line in an HTTP record: Connection: close. This indicates that the connection will be closed after this record is delivered. The closure of an HTTPS connection requires that TLS close the connection with the peer TLS entity on the remote side, which will involve closing the underlying TCP connection. At the TLS level, the proper way to close a connection is for each side to use the TLS alert protocol to send a close_notify alert. TLS implementations must initiate an exchange of closure alerts before closing a connection. A TLS

implementation may, after sending a closure alert, close the connection without waiting for the peer to send its closure alert, generating an “incomplete close”. Note that an implementation that does this may choose to reuse the session. This should only be done when the application knows (typically through detecting HTTP message boundaries) that it has received all the message data that it cares about.
HTTP clients also must be able to cope with a situation in which the underlying TCP connection is terminated without a prior close_notify alert and without a Connection: close indicator. Such a situation could be due to a programming error on the server or a communication error that causes the TCP
connection to drop. However, the unannounced TCP closure could be evidence of some sort of attack. So the HTTPS client should issue some sort of security warning when this occurs.

SECURE SHELL (SSH)

Secure Shell (SSH) is a protocol for secure network communications designed to be relatively simple and inexpensive to implement. The initial version, SSH1 was focused on providing a secure remote logon facility to replace TELNET and other remote logon schemes that provided no security. SSH also provides a more general client/server capability and can be used for such network functions as file transfer and e-mail. A new version, SSH2, fixes a number of security flaws in the original scheme.

SSH2 is documented as a proposed standard in IETF RFCs 4250
through 4256. SSH client and server applications are widely
available for most operating systems.

It has become the method of choice for remote login and X tunneling and is rapidly becomingone of the most pervasive applications for encryption technology outside of embedded systems.

SSH is organized as three protocols that typically run on top of TCP (Figure 16.8):

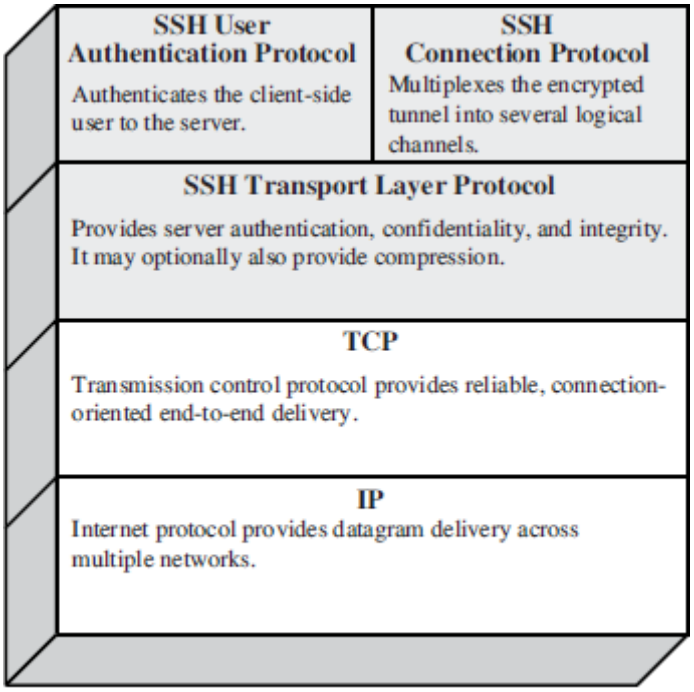


Figure 16.8 SSH Protocol Stack

1) Transport Layer Protocol: Provides server authentication, data confidentiality,and data integrity with forward secrecy (i.e., if a key is compromised during one session, the knowledge does not affect the security of earlier sessions).The transport layer may optionally provide compression..

Once the connection is established, the client and server exchange data, referred to as packets, inthe data field of a TCP segment. Each packet is in the following format (Figure 16.10).

- **Packet length:** Length of the packet in bytes, not including the packet length and MAC fields.
- **Padding length:** Length of the random padding field.
- **Payload:** Useful contents of the packet. Prior to algorithm negotiation, this field is uncompressed. If compression is negotiated, then in subsequent packets, this field is compressed.
- **Random padding:** Once an encryption algorithm has been negotiated this field is added.
- **Message authentication code (MAC):** If message authentication has been negotiated, thisfield contains the MAC value.

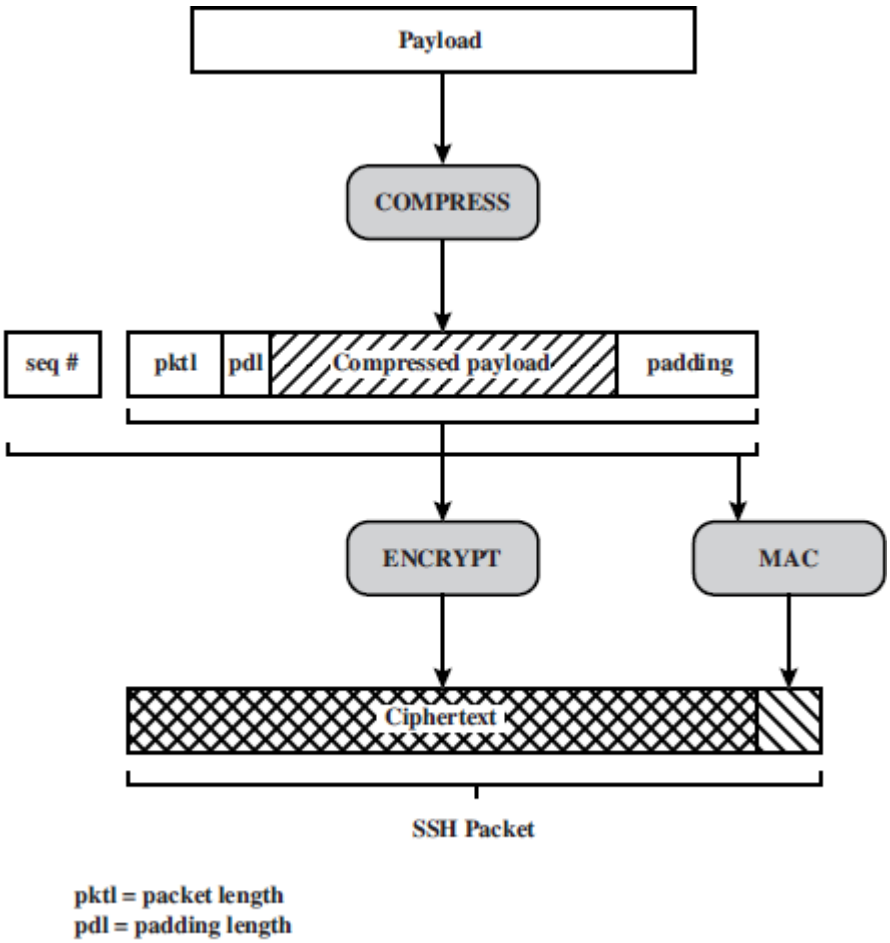


Figure 16.10 SSH Transport Layer Protocol Packet Formation

2) User Authentication Protocol: Authenticates the user to the server.

Authentication Methods: The server may require one or more of the following authenticationmethods

- **Public key:** The details of this method depend on the public-key algorithm chosen. In essence, the client sends a message to the server that contains the client's public key, with themessage signed by the client's private key. When the server receives this message, it checks whether the supplied key is acceptable for authentication and, if so, it checks whether the signature is correct.

• **Password:** The client sends a message containing a plaintext password, which is protected by encryption by the Transport Layer Protocol.

• **host based:** Authentication is performed on the client's host rather than the client itself

3) Connection Protocol: Multiplexes multiple logical communications channels

over a single, underlying SSH connection.

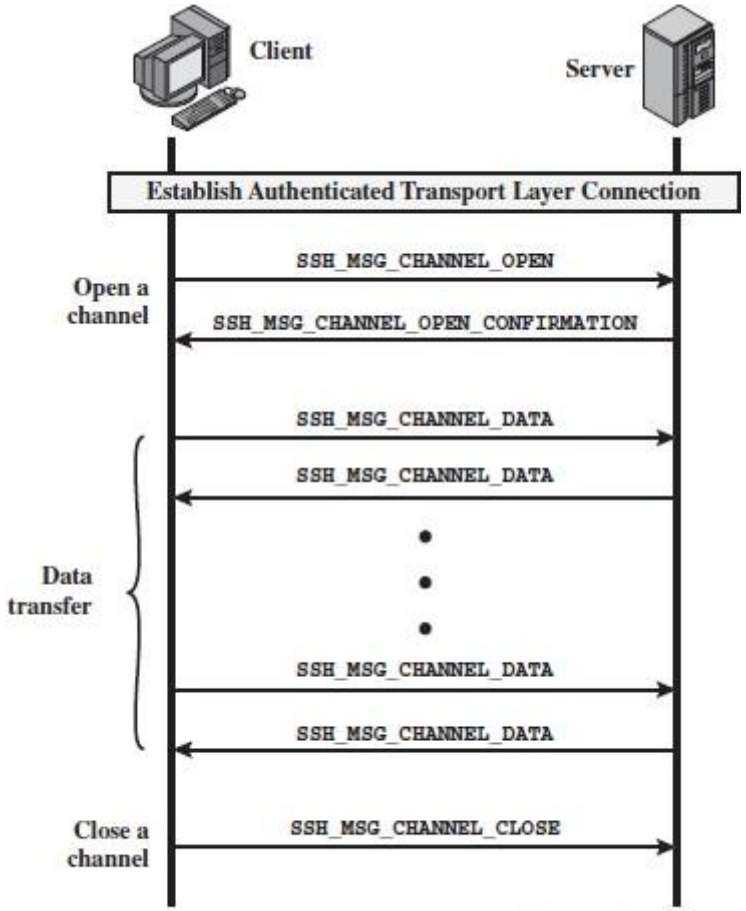


Figure 16.11 Example SSH Connection Protocol Message Exchange

SSH Connection Protocol

❑ Channel Types

- Four channel types are recognized in the SSH Connection Protocol specification.
- **Session:**
  - Remote execution of a program
  - Ex: File transfer, e-mail, system command or some built-in system.
  - Once channel is open subsequent requests are used to start the program

- **X11:**
  - Refers as X windows system
  - Graphics user interface (GUI) for networked computers
  - Application run on server but to be displayed on desktop machine.

- **forwarded-tcpip:**
  - Refers as *remote port forwarding*.

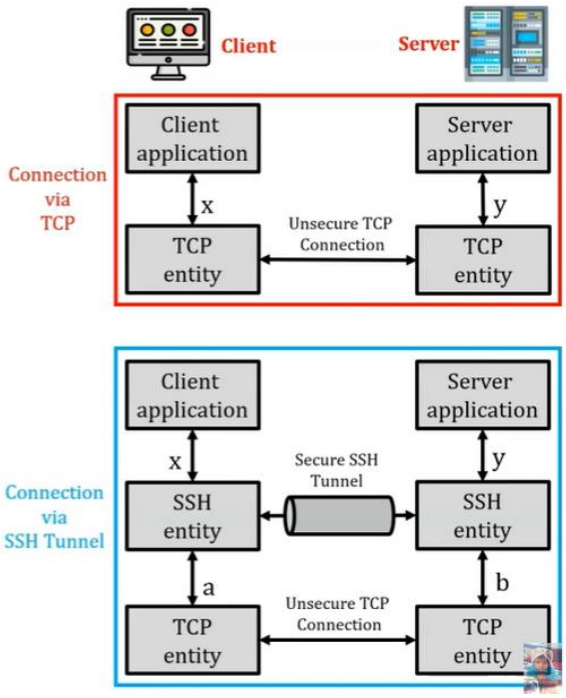
- **direct-tcpip:**

  Refers as *local port forwarding*.

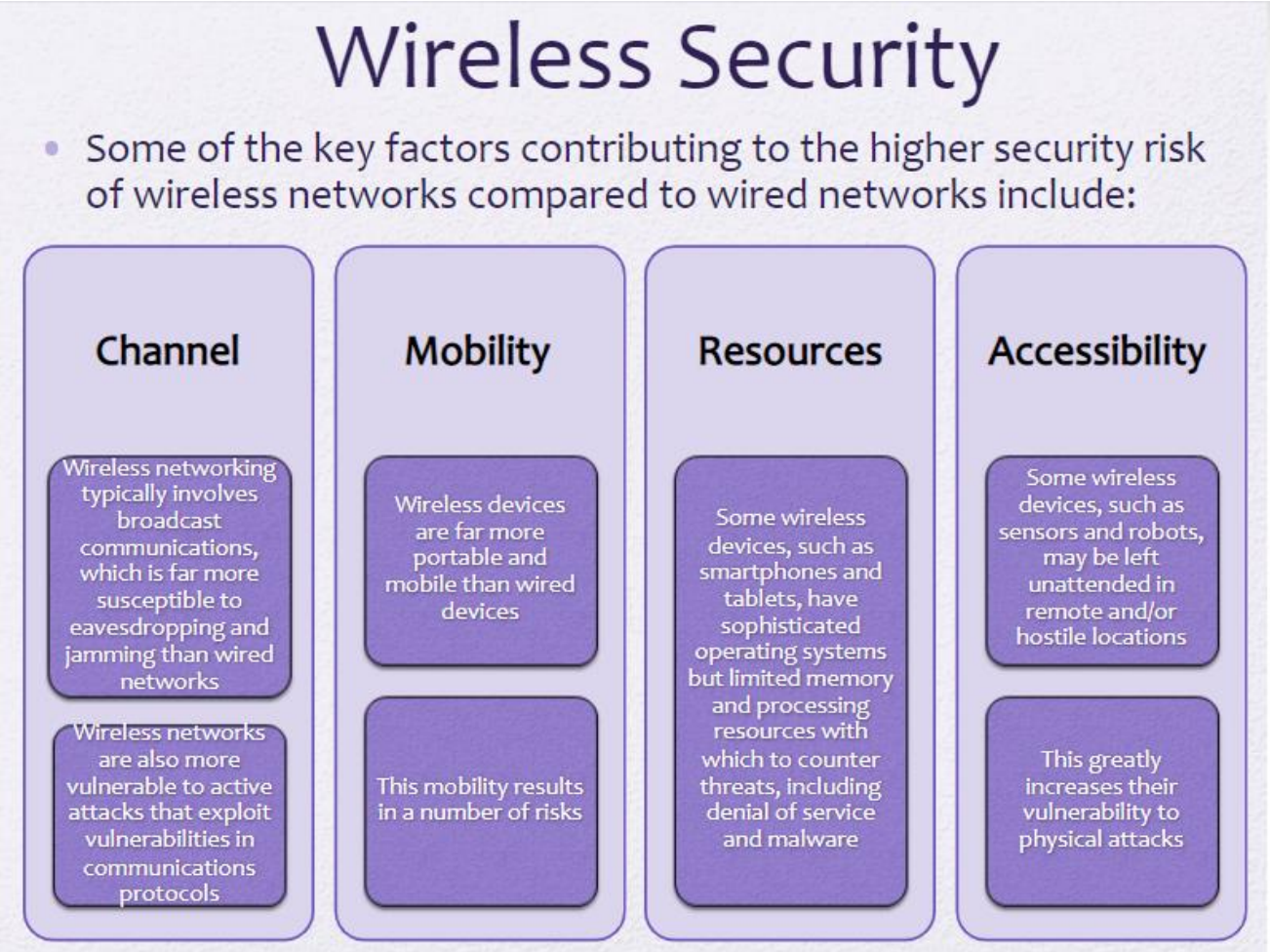
SSH Connection Protocol

❑ Port Forwarding

- Most useful features of SSH
- Port is an identifier of a TCP user.
- Port forwarding ability to convert any TCP insecure connection to secure SSH connection
- Also referred as *SSH tunneling*.
- SSH supports two types of port forwarding:
  - *Local port forwarding*
  - *Remote port forwarding*.



Wireless Security



Securing Wireless Transmissions

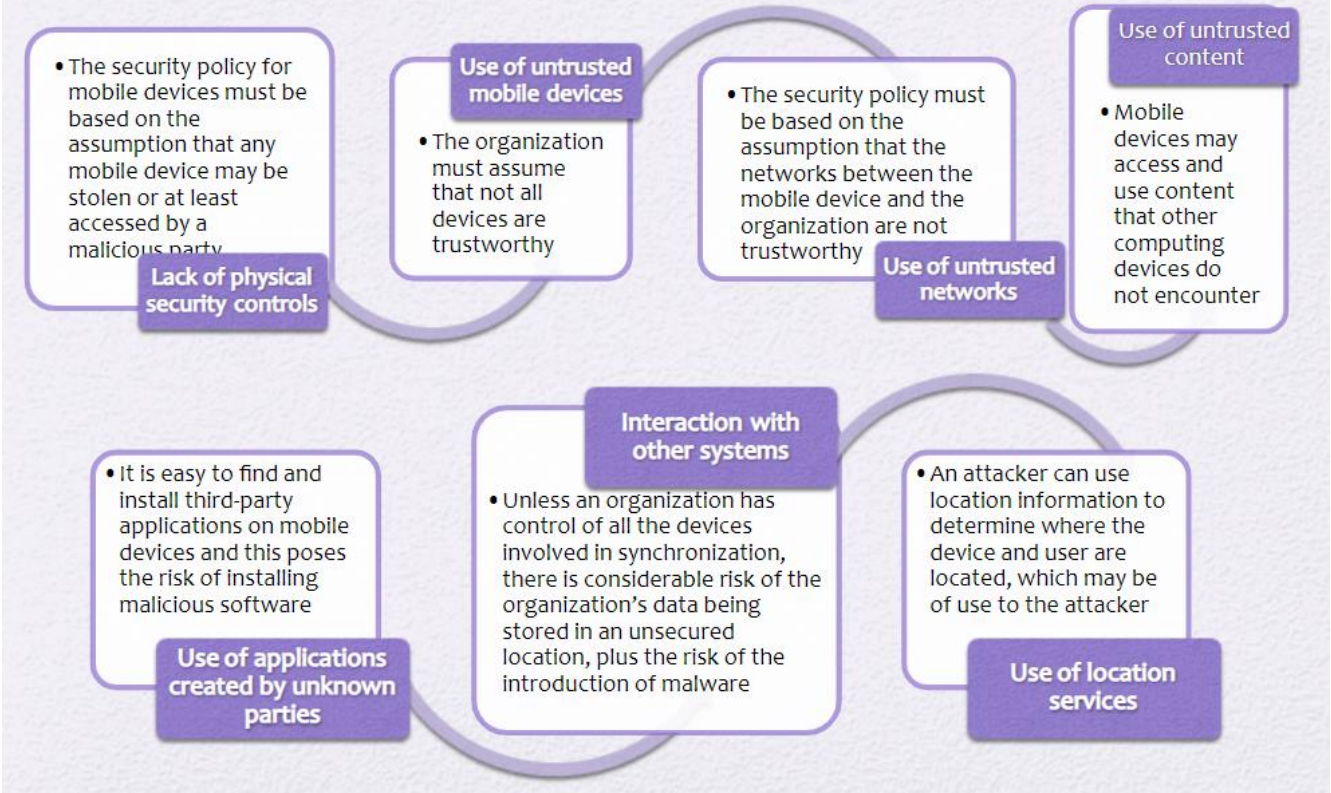
- The principal threats to wireless transmission are eavesdropping, altering or inserting messages, and disruption
- To deal with eavesdropping, two types of countermeasures are appropriate:
  - Signal-hiding techniques
    - Turn off SSID broadcasting by wireless access points
    - Assign cryptic names to SSIDs
    - Reduce signal strength to the lowest level that still provides requisite coverage
    - Locate wireless access points in the interior of the building, away from windows and exterior walls
  - Encryption
    - Is effective against eavesdropping to the extent that the encryption keys are secured

Mobile Device Security

- Mobile devices have become an essential element for organizations as part of the overall network infrastructure
- Prior to the widespread use of smartphones, network security was based upon clearly defined perimeters that separated trusted internal networks from the untrusted Internet
- Due to massive changes, an organization's networks must now accommodate:
  - Growing use of new devices
  - Cloud-based applications
  - De-perimeterization
  - External business requirements

Security Threats

- Major security concerns for mobile devices:



IEEE 802.11 Wireless LAN Overview

IEEE 802 is a committee that has developed standards for a wide range of local area networks (LANs). In 1990, the IEEE 802 Committee formed a new working group, IEEE 802.11, with a charter to develop a protocol and transmission specifications for wireless LANs (WLANs). Since that time, the demand for WLANs, at different frequencies and data rates, has exploded. Keeping pace with this demand, the IEEE 802.11 working group has issued an ever-expanding list of standards.

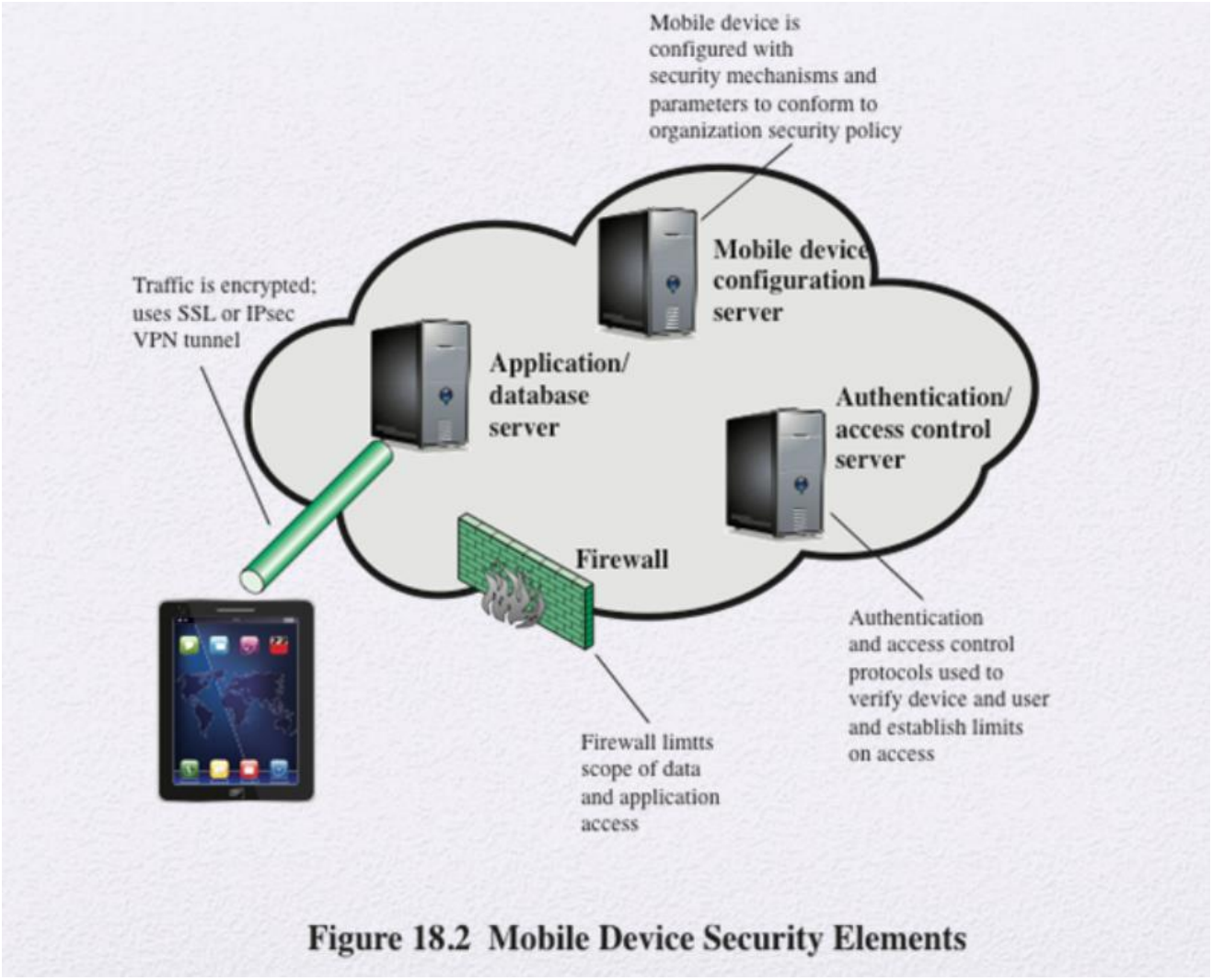


Figure 18.2 Mobile Device Security Elements

Table 18.1 IEEE 802.11 Terminology

Access point (AP)	Any entity that has station functionality and provides access to the distribution system via the wireless medium for associated stations.
Basic service set (BSS)	A set of stations controlled by a single coordination function.
Coordination function	The logical function that determines when a station operating within a BSS is permitted to transmit and may be able to receive PDUs.
Distribution system (DS)	A system used to interconnect a set of BSSs and integrated LANs to create an ESS.
Extended service set (ESS)	A set of one or more interconnected BSSs and integrated LANs that appear as a single BSS to the LLC layer at any station associated with one of these BSSs.
MAC protocol data unit (MPDU)	The unit of data exchanged between two peer MAC entities using the services of the physical layer.
MAC service data unit (MSDU)	Information that is delivered as a unit between MAC users.
Station	Any device that contains an IEEE 802.11 conformant MAC and physical layer.

Wi-Fi Alliance

• 802.11b first broadly accepted standard

• Wireless Ethernet Compatibility Alliance (WECA) industry consortium formed 1999

– to assist interoperability of products

– renamed Wi-Fi (Wireless Fidelity) Alliance

– created a test suite to certify interoperability

– initially for 802.11b, later extended to 802.11g

– concerned with a range of WLANs markets, including enterprise, home, and hot spots

IEEE 802 Protocol Architecture

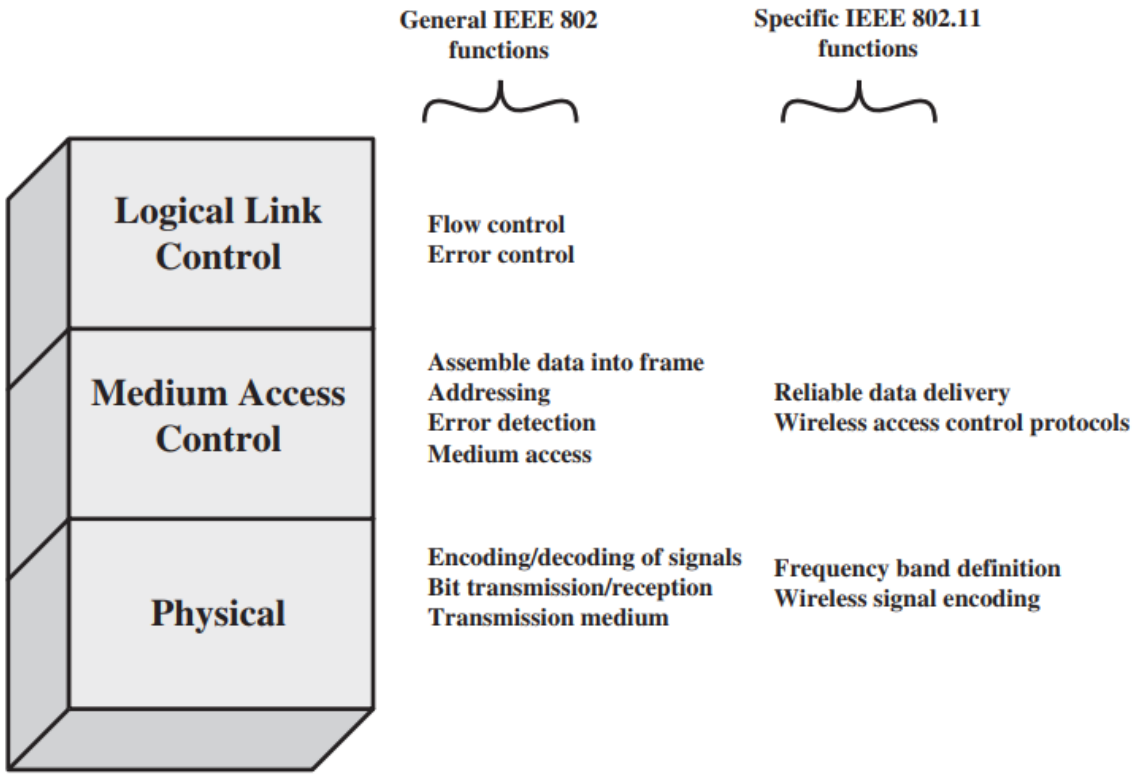


Figure 18.3 IEEE 802.11 Protocol Stack

IEEE 802.11 standards are defined within the structure of a layered set of protocols. This structure, used for all IEEE 802 standards, is illustrated in Figure. The lowest layer of the IEEE 802 reference model is the physical layer, which includes such functions as encoding/decoding of signals and bit transmission/reception. In addition, the physical layer includes a specification of the transmission medium. In the case of IEEE 802.11, the physical layer also defines frequency bands and antenna characteristics.

The media access control (MAC) layer, which controls access to the transmission medium to provide an orderly and efficient use of that capacity. The MAC layer receives data from a higher layer protocol, typically the Logical Link Control (LLC) layer, in the form of a block of data known as the MAC service data unit (MSDU). The exact format of the MPDU differs somewhat for the various MAC protocols in use.

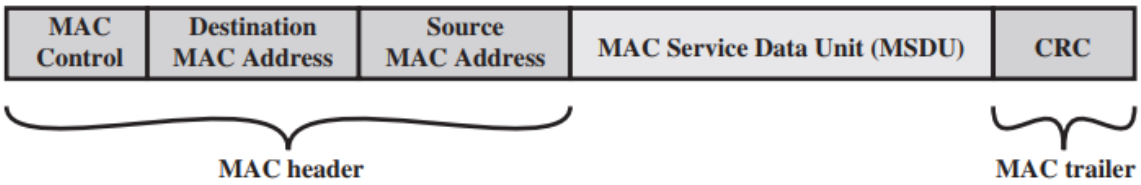


Figure 18.4 General IEEE 802 MPDU Format

In most data link control protocols, the data link protocol entity is responsible not only for detecting errors using the CRC, but for recovering from those errors by

retransmitting damaged frames. In the LAN protocol architecture, these two functions are split between the MAC and LLC layers. The MAC layer is responsible for detecting errors and discarding any frames that contain errors. The LLC layer optionally keeps track of which frames have been successfully received and retransmits unsuccessful frames.

IEEE 802.11 Network Components and Architectural Model

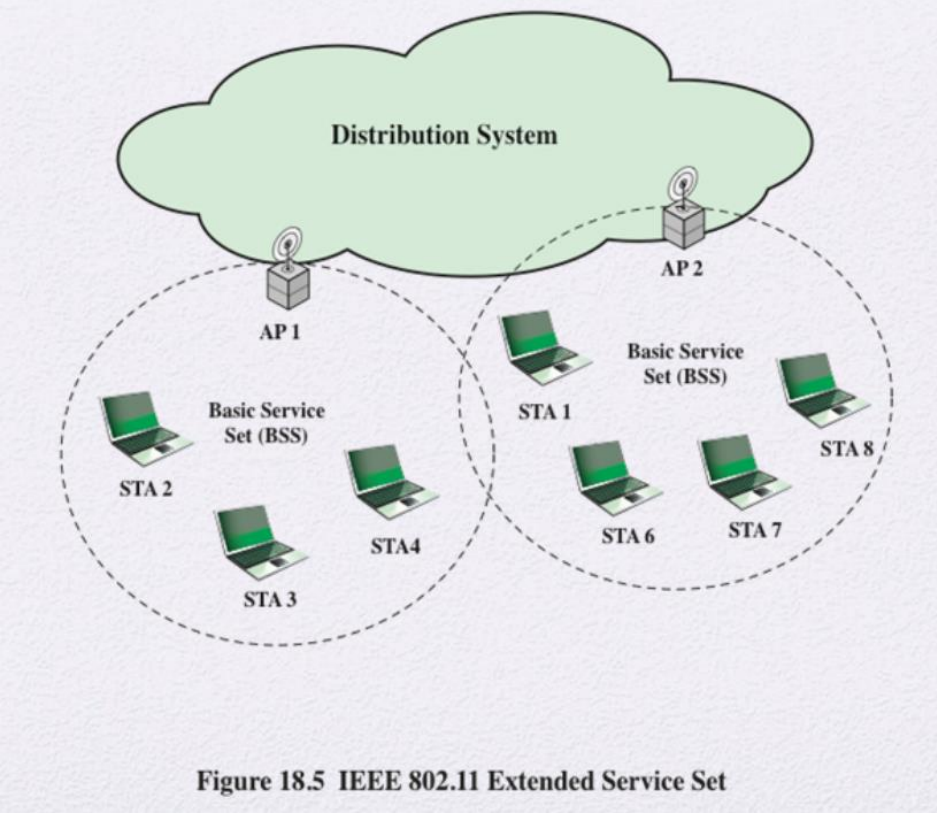


Figure 18.5 IEEE 802.11 Extended Service Set

Figure illustrates the model developed by the 802.11 working group. The smallest building block of a wireless LAN is a basic service set (BSS), which consists of wireless stations executing the same MAC protocol and competing for access to the same shared wireless medium.

A BSS may be isolated or it may connect to a backbone distribution system (DS) through an access point (AP). The AP functions as a bridge and a relay point. In a BSS, client stations do not communicate directly with one another. Rather the MAC frame is first sent from the originating station to the AP, and then from the AP to the destination station. Similarly, a MAC frame from a station in the BSS to a remote station is sent from the local station to the AP and then relayed by the AP over the DS on its way to the destination station. The BSS generally corresponds to what is referred to as a cell.

The DS can be a switch, a wired network, or a wireless network. When all the stations in the BSS are mobile stations that communicate directly with one another, not using an AP, the BSS is called an independent BSS (IBSS). An IBSS is typically an ad hoc network. In an IBSS, the stations all communicate directly, and no AP is involved. A simple configuration is shown in above Figure 5.3, in which each station belongs to a single BSS; that is, each station is within wireless range only of other stations within the same BSS. It is also possible for two BSSs to overlap geographically, so that a single station could participate in more than one BSS. Further, the association between a station and a BSS is dynamic. Stations may turn

off, come within range, and go out of range. An extended service set (ESS) consists of two or more basic service sets interconnected by a distribution system. The extended service set appears as a single logical LAN to the logical link control (LLC) level.

IEEE 802.11 Services

Table 18.2 IEEE 802.11 Services

Service	Provider	Used to support
Association	Distribution system	MSDU delivery
Authentication	Station	LAN access and security
Deauthentication	Station	LAN access and security
Disassociation	Distribution system	MSDU delivery
Distribution	Distribution system	MSDU delivery
Integration	Distribution system	MSDU delivery
MSDU delivery	Station	MSDU delivery
Privacy	Station	LAN access and security
Reassociation	Distribution system	MSDU delivery

IEEE 802.11 defines nine services that need to be provided by the wireless LAN to achieve functionality equivalent to that which is inherent to wired LANs. Table 5.1 lists the services & notes two ways of categorizing them.

1. The service provider can be either the station or the DS. Station services are implemented in every 802.11 station, including AP stations. Distribution services are provided between BSSs; these may be implemented in an AP or in another special-purpose device attached to the distribution system.

2. Three of the services are used to control IEEE 802.11 LAN access and confidentiality. Six of the services are used to support delivery of MSDUs between stations. If the MSDU is too large to be transmitted in a single MPDU, it may be fragmented and transmitted in a series of MPDUs.
MSDU delivery, which is the basic service, has already been mentioned. Distribution is the primary service used by stations to exchange MPDUs when the MPDUs must traverse the DS to get from a station in one BSS to a station in another BSS. Integration enables transfer of data between a station on an IEEE 802.11 LAN and a station on an integrated (wired) IEEE 802.x LAN. To deliver a message within a DS, the distribution service needs to know where the destination station is located. Association establishes an initial association between a station and an AP. Reassociation enables an established association to be transferred from one AP toanother, allowing a mobile station to move from one BSS to another. Disassociation is a notification from either a station or an AP that an existing association is terminated.

IEEE 802.11 Wireless LAN Security :

The differences between wired and wireless LANs (in that wireless traffic can be monitored by any radio in range, and need not be physically connected) suggest the increased need for robust security services and mechanisms for wireless LANs.
The original 802.11 specification included a set of security features for privacy and

authentication that were quite weak. For privacy, 802.11 defined the Wired Equivalent Privacy (WEP) algorithm. The privacy portion of the 802.11 standard contained major weaknesses. Subsequent to the development of WEP, the 802.11i task group has developed a set of capabilities to address the WLAN security issues. In order to accelerate the introduction of strong security into WLANs, the Wi-Fi Alliance promulgated Wi-Fi Protected Access (WPA) as a Wi-Fi standard. WPA is a set of security mechanisms that eliminates most 802.11 security issues and was based on the current state of the 802.11i standard.

The final form of the 802.11i standard is referred to as Robust Security Network (RSN). The Wi-Fi Alliance certifies vendors in compliance with the full 802.11i specification under the WPA 2 program.

802.11i Phases of Operation

The operation of an IEEE 802.11i RSN can be broken down into five distinct phases of operation, as shown in Figure 5.5. One new component is the authentication server (AS). The five phase are:

• Discovery: An AP uses messages called Beacons and Probe Responses to advertise its IEEE802.11i security policy. The STA uses these to identify an AP for a WLAN with which it wishes to communicate. The STA associates with the AP, which it uses to select the cipher suite and authentication mechanism when the Beacons and Probe Responses present a choice.

• Authentication: During this phase, the STA and AS prove their identities to each other. The AP blocks non-authentication traffic between the STA and AS until the authentication transaction is successful. The AP does not participate in the authentication transaction other than forwarding traffic between the STA and AS.

• Key generation and distribution: The AP and the STA perform several operations that cause cryptographic keys to be generated and placed on the AP and the STA. Frames are exchanged between the AP and STA only

• Protected data transfer: Frames are exchanged between the STA and the end station through the AP. As denoted by the shading and the encryption module icon, secure data transfer occurs between the STA and the AP only; security is not provided end-to-end.

• Connection termination: The AP and STA exchange frames. During this phase, the secure connection is torn down and the connection is restored to the original state.

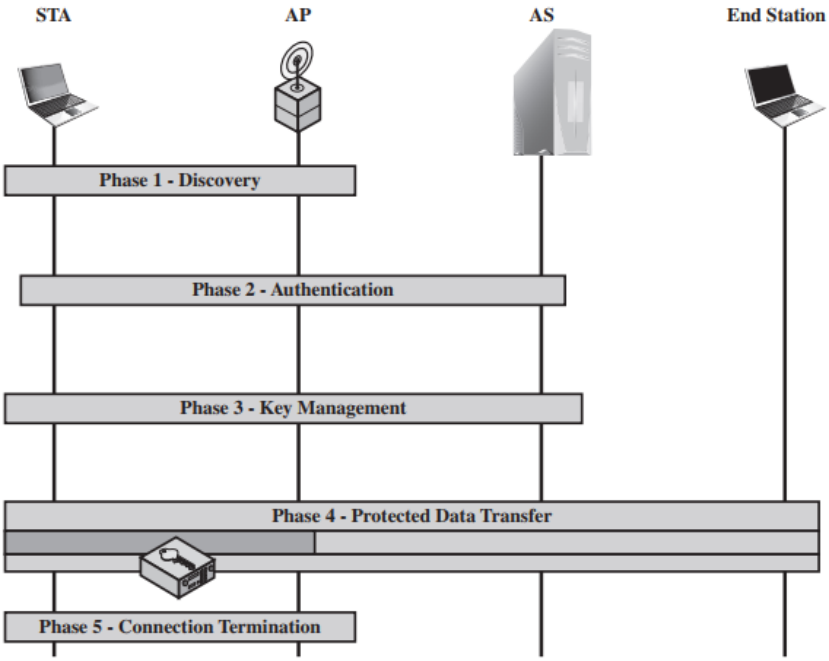


Figure 18.7 IEEE 802.11i Phases of Operation

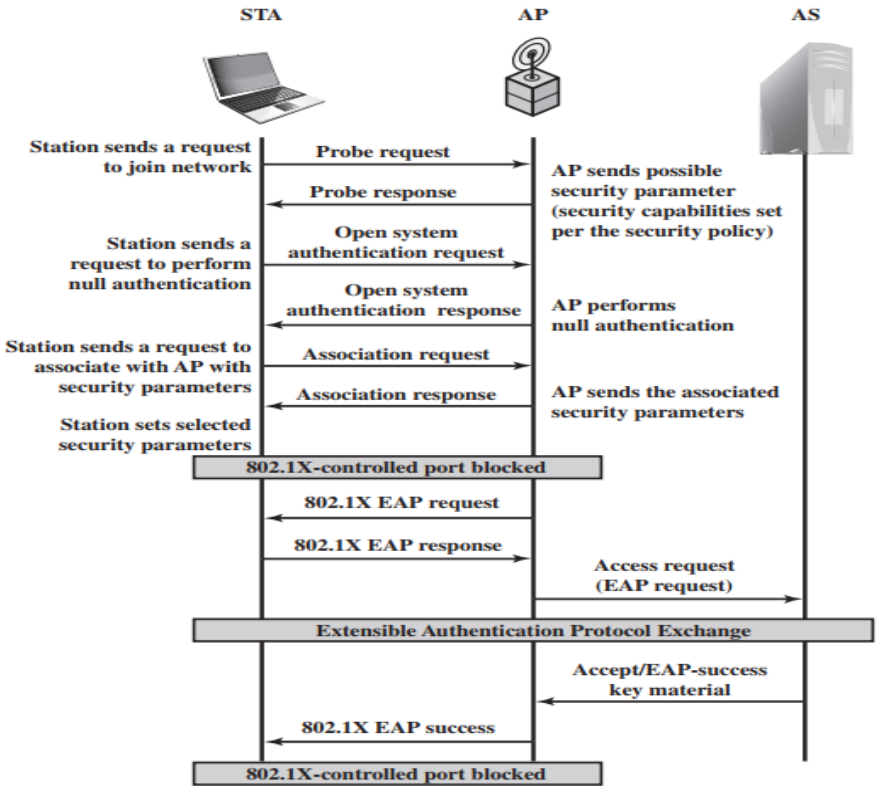


Figure 18.8 IEEE 802.11i Phases of Operation: Capability Discovery, Authentication, and Association

Fig IEEE 802.11i Phases of Operation: Capability Discovery, Authentication & Association
The purpose of this phase is for an STA and an AP to recognize each other, agree on a set of security capabilities, and establish an association for future communication using those security capabilities (Confidentiality and MPDU integrity protocols for protecting

unicast traffic,Authentication method, Cryptography key management approach).

Confidentiality and integrity protocols for protecting multicast/broadcast traffic are dictated by the AP, since all STAs in a multicast group must use the same protocols and ciphers. The specification of a protocol, along with the chosen key length (if variable) is known as a cipher suite. The options for the confidentiality and integrity cipher suite are as follows: WEP, with either a 40-bit or 104-bit key (for backward compatibility), TKIP, CCMP, vendor-specific methods. The options for the authentication and key management (AKM) suite are: IEEE 802.1X, pre-shared key, vendor-specific methods).

The discovery phase consists of three exchanges: Network and security capability discovery, Open system authentication, and Association.

The authentication phase enables mutual authentication between an STA and an authentication server (AS) located in the DS. Authentication is designed to allow only authorized stations to use the network and to provide the STA with assurance that it is communicating with a legitimate network. The lower part of above Figure 5.6 shows the IEEE 802.11 MPDU exchange for this network. The lower part of above Figure 5.6 shows the IEEE 802.11 MPDU exchange for this phase.

802.11i Key Management Phase

Note from Figure that the AP controlled port is still blocked to general user traffic. Although the authentication is successful, the ports remain blocked until the temporal keys are installed in the STA and AP, which occurs during the 4-Way Handshake. During the key management phase, a variety of cryptographic keys are generated and distributed to STAs. There are two types of keys: pairwise keys, used for communication between an STA and an AP; and group keys, for multicast communication. Figure 5.8 shows the two key hierarchies. Pairwise keys are used for communication between a pair of devices, typically between an STA and an AP. These keys form a hierarchy, beginning with a master key from which other keys are derived dynamically and used for a limited period of time. A pre-shared key (PSK) is a secret key shared by the AP and a STA, and installed in some fashion outside the scope of IEEE 802.11i.

The other alternative is the master session key (MSK), also known as the AAAK, which is generated using the IEEE 802.1X protocol during the authentication phase, as described previously. The pairwise master key (PMK) is derived from the master key as follows:

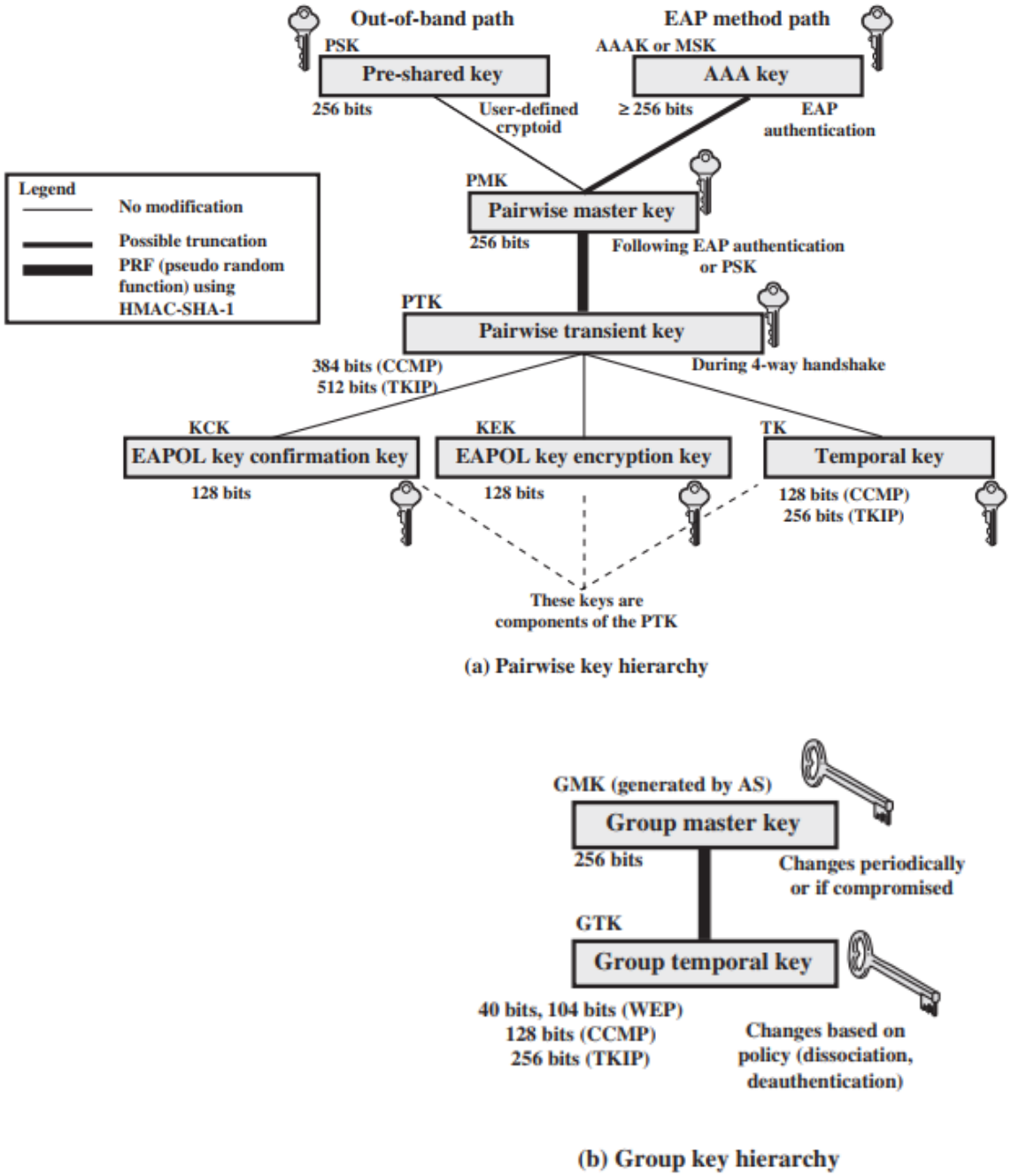


Figure 18.9 IEEE 802.11i Key Hierarchies

If a PSK is used, then the PSK is used as the PMK; if a MSK is used, then the PMK is derived from the MSK by truncation (if necessary). By the end of the authentication phase (on EAP Success message), both the AP and the STA have a copy of their shared PMK. The PMK is used to generate the pairwise

transient key (PTK), which in fact consists of three keys to be used for communication between an STA and AP after they have mutually authenticated. To derive the PTK, the PMK, the MAC addresses of the STA and AP, and nonces generated when needed are all input to the HMAC-SHA-1 function. Group keys are used for multicast communication when one STA sends MPDU's to multiple STAs.

802.11i Key Management Phase

The upper part of Figure shows the MPDU exchange for distributing pairwise keys. This exchange is known as the 4-way handshake. The STA and SP use this handshake to confirm the existence of the PMK, verify the selection of the cipher suite, and derive a fresh PTK for the following data session.

For group key distribution, the AP generates a GTK and distributes it to each STA in a multicast group

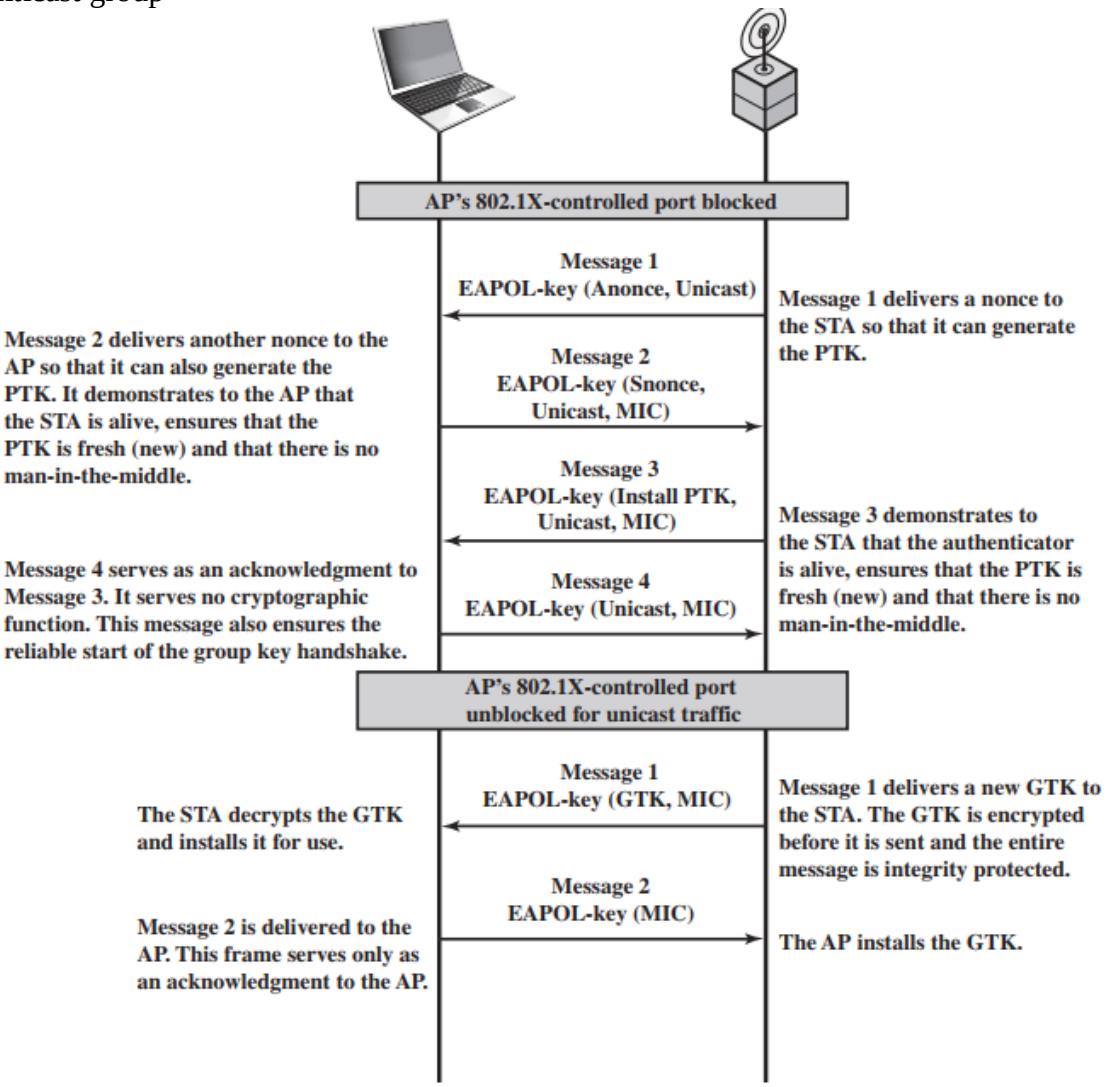


Figure 18.10 IEEE 802.11i Phases of Operation: Four-Way Handshake and Group Key Handshake

UNIT 5
CRYPTOGRAPHY AND NETWORK SECURITY

ELECTRONIC MAIL SECURITY

The protection of email from unauthorized access and inspection is known as **electronicprivacy**. There are mainly two methods for proving security for electronic mails

- Pretty Good Privacy
- S/MIME

Pretty Good Privacy:

In virtually all distributed environments, electronic mail is the most heavily used network basedapplication.

Introduction:

- PGP is data encryption and decryption computer program that provides privacy(Confidentiality) and authentication for data communication.
- It was created by Phil Zimmermann in 1991

Use of PGP:

- It is used in Electronic mail
- File storage applications.
- PGP is an open-source, freely available software package for e-mail security. It provides **authentication** through the use of digital signature, **confidentiality** through the use of symmetric block encryption, **compression** using the ZIP algorithm, and **e-mail compatibility** using the radix-64 encoding scheme.
- PGP incorporates tools for developing a public-key trust model and public-key certificatemanagement

PGP has grown explosively and is now widely used, because of following reasons:

1. It is available free worldwide in versions that run on a variety of platforms, including Windows, UNIX, Macintosh, and many more. In addition, the commercial version satisfies users who want a product that comes with vendor support.
2. It is based on algorithms that have survived extensive public review and are considered extremely secure. Specifically, the package includes RSA, DSS, and Diffie-Hellman for public-key encryption; CAST-128, IDEA, and 3DES for symmetric encryption; andSHA-1 for hash coding.
3. It has a wide range of applicability, from corporations that wish to select and enforce a standardized scheme for encrypting files and messages to individuals who wish to communicate securely with others worldwide over the Internet and other networks.
4. It was not developed by, nor is it controlled by, any governmental or standards organization. For those with an instinctive distrust of "the establishment," this makes PGP attractive.

NOTATIONS

The following symbols are used in PGP

K_s = session key used in symmetric encryption scheme
PR_a = private key of user A, used in public-key encryption scheme
PU_a = public key of user A, used in public-key encryption scheme
EP = public-key encryption
DP = public-key decryption
EC = symmetric encryption
DC = symmetric decryption
H = hash function
|| = concatenation
Z = compression using ZIP algorithm
R64 = conversion to radix 64 ASCII format[1]

PGP SERVICES

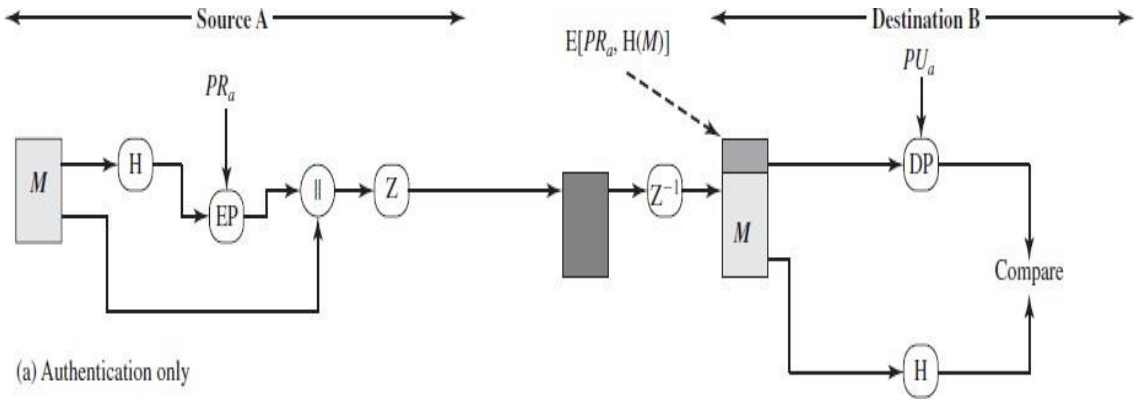
1) authentication
2) confidentiality
3) compression
4) e-mail compatibility

Function	**Algorithms Used**	**Description**
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1.This message digest is encryptedusing DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman or RSA	A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key and included with the message.
Compression	ZIP	A message may be compressed for storage or transmission using ZIP.
E-mail compatibility	Radix-64 conversion	To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix-64 conversion.

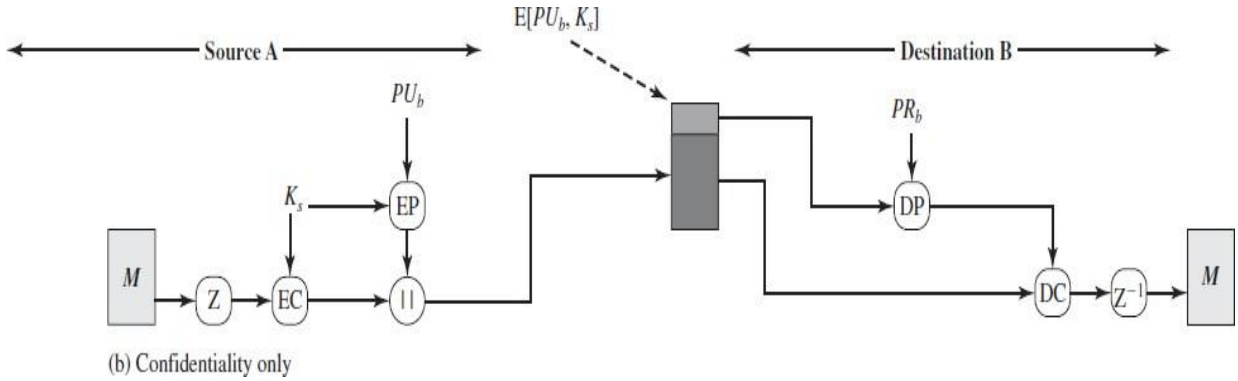
<u>Providing authentication by using PGP</u>:

The sequence steps for providing authentication by using PGP

1. The sender creates a message.
2. SHA-1 is used to generate a 160-bit hash code of the message.
3. The hash code is encrypted with RSA using the sender's private key, and the result isprepended to the message.
4. The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
5. The receiver generates a new hash code for the message and compares it with thedecrypted hash code. If the two match, the message is accepted as authentic.



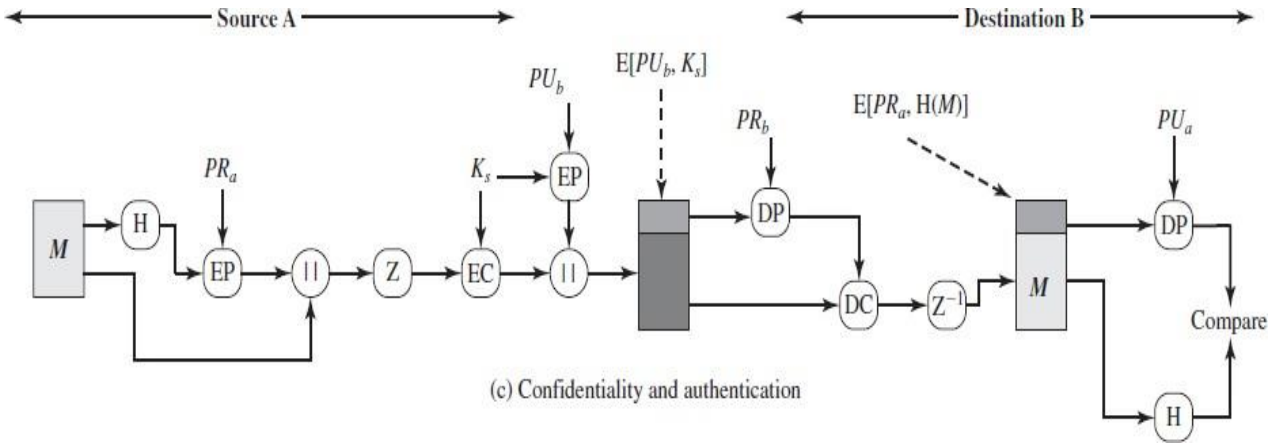
Confidentiality by using PGP



Steps for providing confidentiality:

1. The sender generates a message and a random 128-bit number to be used as a session keyfor this message only.
2. The message is encrypted using CAST-128 (or IDEA or 3DES) with the session key.
3. The session key is encrypted with RSA using the recipient's public key and is prependedto the message.
4. The receiver uses RSA with its private key to decrypt and recover the session key.
5. The session key is used to decrypt the message.

PGP for both authentication and confidentiality:



(c) Confidentiality and authentication

COMPRESSION

As a default, PGP compresses the message after applying the signature but before Encryption.This has the benefit of saving space both for e-mail transmission and for file storage.

The signature is generated before compression for two reasons

- so can store uncompressed message & signature for later verification
- Message encryption is applied after compression to strengthen cryptographic security.Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

PGP uses ZIP compression algorithm

E-MAIL COMPATIBILITY

When PGP is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender's private key). If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one- time symmetric key).Thus, part or the entire resulting block consists of a stream of arbitrary 8-bitoctets.

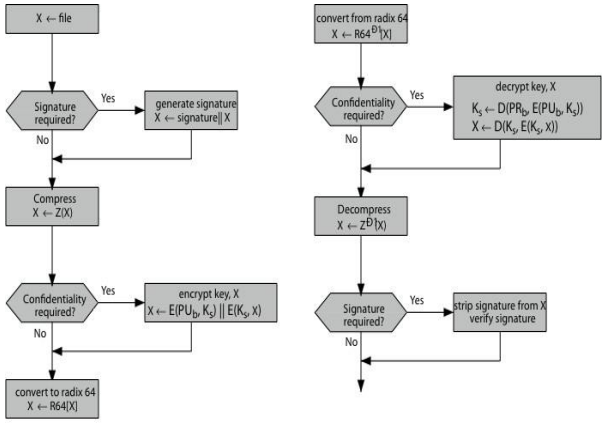
However, many electronic mail systems only permit the use of blocks consisting of ASCII text.To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters.

The scheme used for this purpose is radix-64 algorithm

- It maps 3 bytes to 4 printable chars
- It also appends a CRC to detect transmission errorsPGP also

segments messages if too big

PGP Operation – Summary



(a) Generic Transmission Diagram (from A) (b) Generic Reception Diagram (to B)

S/MIME

- Secure/Multipurpose Internet Mail Extension is a security enhancement to the MIMEinternet email standard.
- S/MIME is for industry standard for commercial and organizational use.
- It defined in number of documents that is RFC 2630, RFC 2632, RFC 2633

E-mail format standard, RFC 822, which is still in common use. The most recent version of thisformat specification is RFC 5322 (Internet Message Format).

RFC 5322 (Internet Message Format).

RFC 5322 defines a format for text messages that are sent using electronic mail. It has been thestandard for Internet-based text mail messages and remains in common use.

Message Structure

- A message consists of some number of header lines (*the header*) followed by unrestrictedtext (*the body*).
- A header line usually consists of a keyword, followed by a colon, followed bythe keyword's arguments

Multipurpose Internet Mail Extensions:

Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework thatis intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP).

The following are limitations of the SMTP/5322 scheme.

1. SMTP cannot transmit executable files or other binary objects. A number of schemes arein use for converting binary files into a text form that can be used by SMTP mail systems.
2. SMTP cannot transmit text data that includes national language characters, because theseare represented by 8-bit codes, and SMTP is limited to 7-bit ASCII.

3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not usea consistent set of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle non-textual dataincluded in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards definedin RFC 821. Common problems include:
    a. Deletion, addition, or reordering of carriage return and linefeed
    b. Truncating or wrapping lines longer than 76 characters
    c. Removal of trailing white space (tab and space characters)
    d. Padding of lines in a message to the same length
    e. Conversion of tab characters into multiple space characters

MIME is intended to resolve these problems in a manner that is compatible with existing RFC5322 implementations. The specification is provided in RFCs 2045 through 2049.

MIME has 5 header fields

The five header fields defined in MIME are

1. **MIME-Version**: Must have the parameter value 1.0.
2. **Content-Type**: Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user
3. **Content-Transfer-Encoding**: Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
4. **Content-ID**: Used to identify MIME entities uniquely in multiple contexts.
5. **Content-Description**: A text description of the object with the body; this is useful whenthe object is not readable (e.g., audio data).

Mime Content Types

Describes the data contained in the body with sufficient detail

Table 18.3 MIME Content Types

Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript format.
	octet-stream	General binary data consisting of 8-bit bytes.

S/MIME Functionality

In terms of general functionality, S/MIME is very similar to PGP. Both offer the ability to signand/or encrypt messages. In this subsection, we briefly summarize S/MIME capability.

S/MIME provides the following functions.

- **Enveloped data:** This consists of encrypted content of any type and encrypted-content encryption keys for one or more recipients.
- **Signed data:** A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.
    - encoded (message + signed digest)

- **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64.As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.

- clear text message + encoded (signed digest) **Signed and enveloped data:** Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.
    - nesting of signed & encrypted entities

CRYPTOGRAPHIC ALGORITHMS

S/MIME uses the following terminology taken from RFC 2119 (Key Words for use in RFCs to Indicate Requirement Levels) to specify the requirement level:

- **MUST**: The definition is an absolute requirement of the specification. An implementation must include this feature or function to be in conformance with the specification.
- **SHOULD**: There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function.

Table 18.6 Cryptographic Algorithms Used in S/MIME

Function	Requirement
Create a message digest to be used in forming a digital signature.	MUST support SHA-1. Receiver SHOULD support MD5 for backward compatibility.
Encrypt message digest to form a digital signature.	Sending and receiving agents MUST support DSS. Sending agents SHOULD support RSA encryption. Receiving agents SHOULD support verification of RSA signatures with key sizes 512 bits to 1024 bits.
Encrypt session key for transmission with a message.	Sending and receiving agents SHOULD support Diffie-Hellman. Sending and receiving agents MUST support RSA encryption with key sizes 512 bits to 1024 bits.
Encrypt message for transmission with a one-time session key.	Sending and receiving agents MUST support encryption with tripleDES. Sending agents SHOULD support encryption with AES. Sending agents SHOULD support encryption with RC2/40.
Create a message authentication code.	Receiving agents MUST support HMAC with SHA-1. Sending agents SHOULD support HMAC with SHA-1.

S/MIME incorporates three public-key algorithms. The Digital Signature Standard (DSS) is thepreferred algorithm for digital signature.

IP SECURITY

IP-level security encompasses three functional areas: authentication, confidentiality, and key management. The authentication mechanism assures that a received packet was, in fact, transmitted by the party identified as the source in the packet header. In addition, this mechanism assures that the packet has not been altered in transit. The confidentiality facility enables communicating nodes to encrypt messages to prevent eavesdropping by third parties. The key management facility is concerned with the secure exchange of keys.

IP Security Overview:

The IP security capabilities were designed to be used for both with the current IPv4 and thefuture IPv6 protocols.

Applications of IPSec:

IPSec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include the following:

- **Secure branch office connectivity over the Internet:** A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead.
- **Secure remote access over the Internet:** An end user whose system is equipped with IP security protocols can make a local call to an Internet service provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for traveling employees and telecommuters.
- **Establishing extranet and intranet connectivity with partners:** IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.
- **Enhancing electronic commerce security:** Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security.

The principal feature of IPSec that enables it to support these varied applications is that it can encrypt and/or authenticate *all* traffic at the IP level. Thus, all distributed applications, including remote logon, client/server, e-mail, file transfer, Web access, and so on, can be secured.

Figure 1.1 is a typical scenario of IPSec usage. An organization maintains LANs at dispersed locations. Nonsecure IP traffic is conducted on each LAN. For traffic offsite, through some sort of private or public WAN, IPSec protocols are used. These protocols operate in networking devices, such as a router or firewall, that connect each LAN to the outside world. The IPSec networking device will typically encrypt and compress all traffic going into the WAN, and decrypt and decompress traffic coming from the WAN; these operations are transparent to workstations and servers on the LAN. Secure transmission is also possible with individual users who dial into the WAN. Such user workstations must implement the IPSec protocols to provide security.

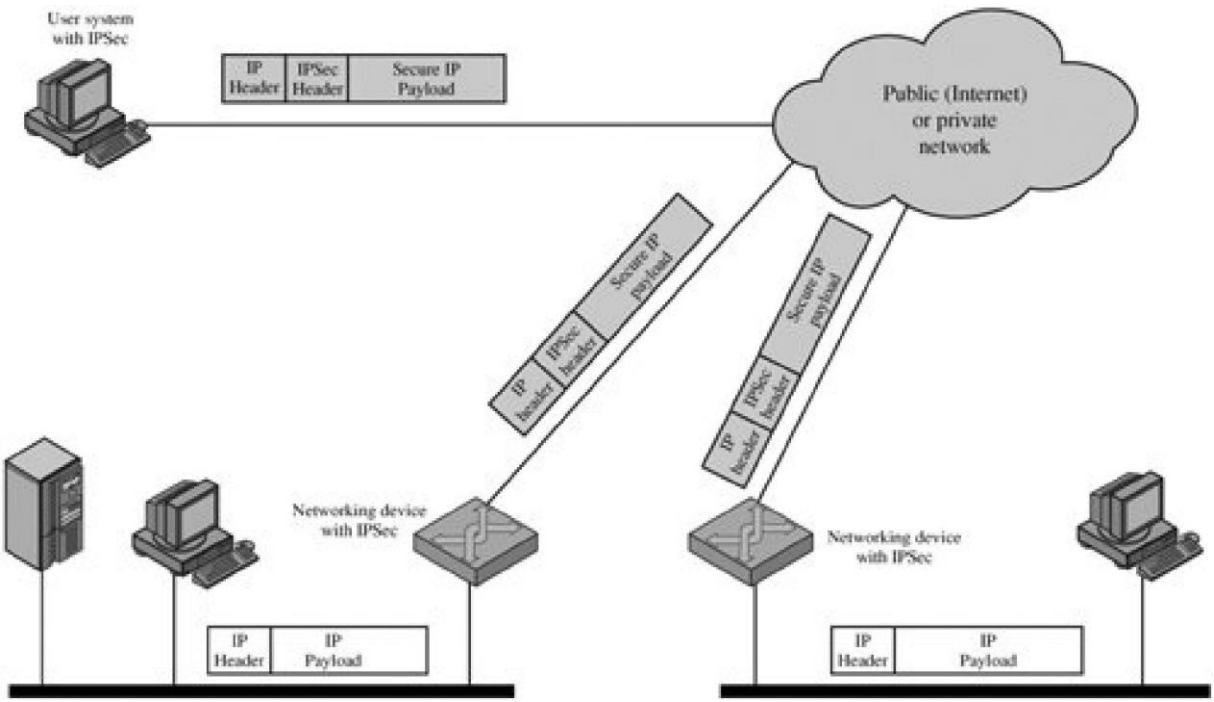


Figure 1.1. An IP Security Scenario

Benefits of IPSec:

The following are the benefits of IPSec:

- When IPSec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.
- IPSec in a firewall is resistant to bypass if all traffic from the outside must use IP, and the firewall is the only means of entrance from the Internet into the organization.
- IPSec is below the transport layer (TCP, UDP) and so is transparent to applications. There is no need to change software on a user or server system when IPSec is implemented in the firewall or router. Even if IPSec is implemented in end systems, upper-layer software, including applications, is not affected.
- IPSec can be transparent to end users. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization.
- IPSec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual subnetwork within an organization for sensitive applications.

Routing Applications:

In addition to supporting end users and protecting premises systems and networks, IPSec can play a vital role in the routing architecture required for internetworking. [HUIT98] lists the following examples of the use of IPSec. IPSec can assure that

- A router advertisement (a new router advertises its presence) comes from an authorized router
- A neighbor advertisement (a router seeks to establish or maintain a neighbor relationship with a router in another routing domain) comes from an authorized router.
- A redirect message comes from the router to which the initial packet was sent.
- A routing update is not forged.

Without such security measures, an opponent can disrupt communications or divert some traffic. Routing protocols such as OSPF should be run on top of security associations between routers that are defined by IPSec.

IP Security Architecture:

The IPSec specification has become quite complex. To get a feel for the overall architecture, we begin with a look at the documents that define IPSec. Then we discuss IPSec services andintroduce the concept of security association.

IPSec Documents:

The IPSec specification consists of numerous documents. The most important of these, issued in November of 1998, are RFCs 2401, 2402, 2406, and 2408:

- RFC 2401: An overview of a security architecture
- RFC 2402: Description of a packet authentication extension to IPv4 and IPv6
- RFC 2406: Description of a packet encryption extension to IPv4 and IPv6
- RFC 2408: Specification of key management capabilities

Support for these features is mandatory for IPv6 and optional for IPv4. In both cases, the security features are implemented as extension headers that follow the main IP header. The extension header for authentication is known as the Authentication header; that for encryption is known as the Encapsulating Security Payload (ESP) header.

In addition to these four RFCs, a number of additional drafts have been published by the IP Security Protocol Working Group set up by the IETF. The documents are divided into seven groups, as depicted in Figure 1.2 (RFC 2401).

- **Architecture:** Covers the general concepts, security requirements, definitions, and mechanisms defining IPSec technology.
- **Encapsulating Security Payload (ESP):** Covers the packet format and general issues related to the use of the ESP for packet encryption and, optionally, authentication.
- **Authentication Header (AH):** Covers the packet format and general issues related tothe use of AH for packet authentication.
- **Encryption Algorithm:** A set of documents that describe how various encryption algorithms are used for ESP.

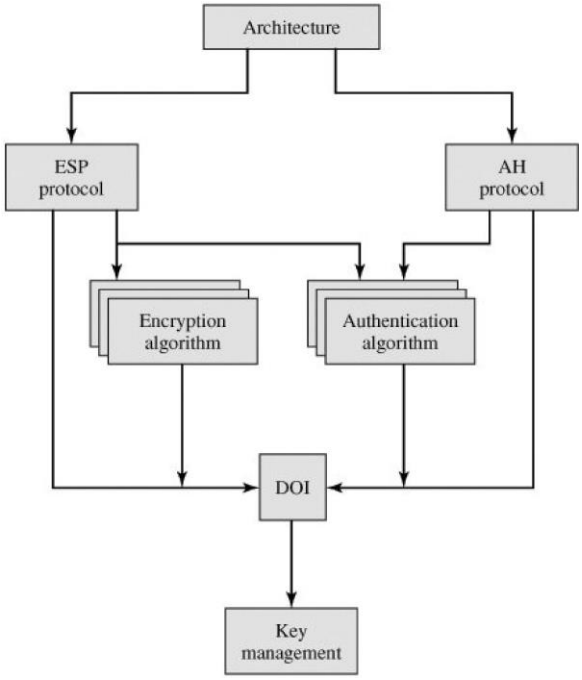


Figure 1.2. IPSec Document Overview

- **Authentication Algorithm:** A set of documents that describe how variousauthentication algorithms are used for AH and for the authentication option of ESP.
- **Key Management:** Documents that describe key management schemes.
- **Domain of Interpretation (DOI):** Contains values needed for the other documents torelate to each other. These include identifiers for approved encryption and authentication algorithms, as well as operational parameters such as key lifetime.

IPSec Services:

IPSec provides security services at the IP layer by enabling a system to select requiredsecurity protocols, determine the algorithm(s) to use for the service(s), and put in place anycryptographic keys required to provide the requested services. Two protocols are used to provide security: an authentication protocol designated by the header of the protocol,Authentication Header (AH); and a

combined encryption/authentication protocol designatedby the format of the packet for that protocol, Encapsulating Security Payload (ESP). The services are

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets (a form of partial sequence integrity)
- Confidentiality (encryption)
- Limited traffic flow confidentiality

Table 1.1 shows which services are provided by the AH and ESP protocols. For ESP, there are two cases: with and without the authentication option. Both AH and ESP are vehicles for access control, based on the distribution of cryptographic keys and the management of traffic flows relative to these security protocols.

Table 1.1. IPSec Services

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✔	✔	✔
Connectionless integrity	✔		✔
Data origin authentication	✔		✔
Rejection of replayed packets	✔	✔	✔
Confidentiality		✔	✔
Limited traffic flow confidentiality		✔	✔

Security Associations:

A key concept that appears in both the authentication and confidentiality mechanisms for IPis the security association (SA). **An association is a one-way relationship between a sender and a receiver that affords security services to the traffic carried on it**. If a peer relationship is needed, for two-way secure exchange, then two security associations are required. Security services are afforded to an SA for the use of AH or ESP, but not both.

A security association is uniquely identified by three parameters:

Security Parameters Index (SPI): A bit string assigned to this SA and having local significance only. The SPI is carried in AH and ESP headers to enable the receiving systemto select the SA under which a received packet will be processed.

IP Destination Address: Currently, only unicast addresses are allowed; this is the address of the destination endpoint of the SA, which may be an end user system or a network system such as a firewall or router.

Security Protocol Identifier: This indicates whether the association is an AH or ESP security association.

Hence, in any IP packet, the security association is uniquely identified by the Destination Address in the IPv4 or IPv6 header and the SPI in the enclosed extension header (AH or ESP).

SA Parameters:

In each IPSec implementation, there is a nominal Security Association Database that defines the parameters associated with each SA. A security association is normally defined by the following parameters:

- **Sequence Number Counter:** A 32-bit value used to generate the Sequence Number field in AH or ESP headers.
- **Sequence Counter Overflow:** A flag indicating whether overflow of the Sequence Number Counter should generate an auditable event and prevent further transmission of packets on this SA (required for all implementations).
- **Anti-Replay Window:** Used to determine whether an inbound AH or ESP packet is areplay.
- **AH Information:** Authentication algorithm, keys, key lifetimes, and related parameters being used with AH (required for AH implementations).
- **ESP Information:** Encryption and authentication algorithm, keys, initialization values, key lifetimes, and related parameters being used with ESP (required for ESP implementations).
- **Lifetime of This Security Association:** A time interval or byte count after which an SA must be replaced with a new SA (and new SPI) or terminated, plus an indicationof which of these actions should occur (required for all implementations).
- **IPSec Protocol Mode:** Tunnel, transport, or wildcard (required for all implementations).
- **Path MTU:** Any observed path maximum transmission unit (maximum size of a packet that can be transmitted without fragmentation) and aging variables (required for all implementations).

The key management mechanism that is used to distribute keys is coupled to the authentication and privacy mechanisms only by way of the Security Parameters Index. Hence, authentication and privacy have been specified independent of any specific key management mechanism.

SA Selectors:

IPSec provides the user with considerable flexibility in the way in which IPSec services are applied to IP traffic. SAs can be combined in a number of ways to yield the desired user configuration. Furthermore, IPSec provides a high degree of granularity in discriminating between traffic that is afforded IPSec protection and traffic that is allowed to bypass IPSec,in the former case relating IP traffic to specific SAs.

The means by which IP traffic is related to specific SAs (or no SA in the case of traffic allowed to bypass IPSec) is the nominal Security Policy Database (SPD). In its simplest form, an SPD contains entries, each of which defines a subset of IP traffic and points to an SA for that traffic. In more complex environments, there may be multiple entries that potentially relate to a single SA or multiple

SAs associated with a single SPD entry. The reader is referred to the relevant IPSec documents for a full discussion.

Each SPD entry is defined by a set of IP and upper-layer protocol field values, called *selectors*. In effect, these selectors are used to filter outgoing traffic in order to map it into a particular SA. Outbound processing obeys the following general sequence for each IP packet:

- Compare the values of the appropriate fields in the packet (the selector fields) againstthe SPD to find a matching SPD entry, which will point to zero or more SAs.
- Determine the SA if any for this packet and its associated SPI.
- Do the required IPSec processing (i.e., AH or ESP processing).

The following selectors determine an SPD entry:

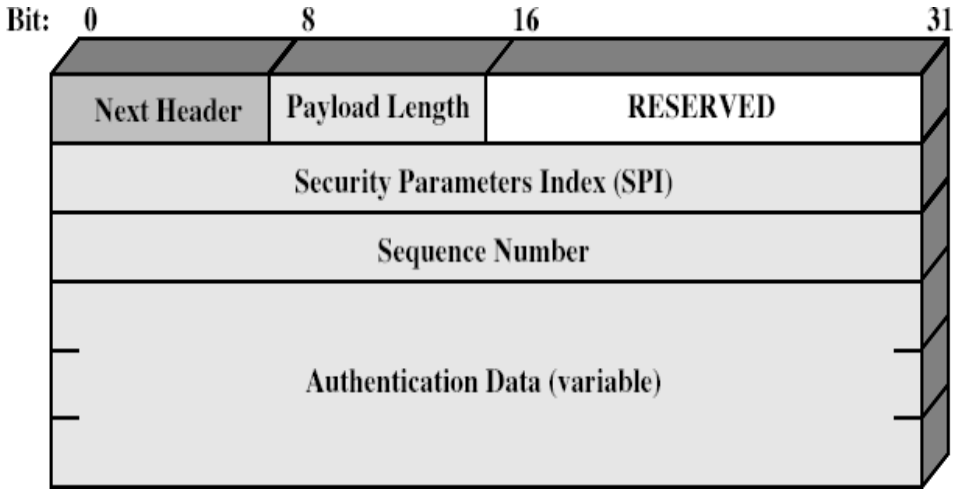
- **Destination IP Address:** This may be a single IP address, an enumerated list orrange of addresses, or a wildcard (mask) address. The latter two are required to support more than one destination system sharing the same SA (e.g., behind a firewall).
- **Source IP Address:** This may be a single IP address, an enumerated list or range of addressee, or a wildcard (mask) address. The latter two are required to support more than one source system sharing the same SA (e.g., behind a firewall).
- **User ID:** A user identifier from the operating system. This is not a field in the IP or upper-layer headers but is available if IPSec is running on the same operating system as the user.
- **Data Sensitivity Level:** Used for systems providing information flow security (e.g., Secret or Unclassified).
- **Transport Layer Protocol:** Obtained from the IPv4 Protocol or IPv6 Next Header field. This may be an individual protocol number, a list of protocol numbers, or a range of protocol numbers.
- **Source and Destination Ports:** These may be individual TCP or UDP port values, an enumerated list of ports, or a wildcard port.

Authentication Header:

The Authentication Header provides support for data integrity and authentication of IP packets. The data integrity feature ensures that undetected modification to a packet's content in transit is not possible. The authentication feature enables an end system or network device to authenticate the user or application and filter traffic accordingly; it also prevents the address spoofing attacks observed in today's Internet. The AH also guards against the replay attack.

Authentication is based on the use of a message authentication code (MAC), hence the two parties must share a secret key.

Figure 1.3 IPSec Authentication Header



The Authentication Header consists of the following fields (Figure 1.3):

- **Next Header (8 bits):** Identifies the type of header immediately following thisheader.
- **Payload Length (8 bits):** Length of Authentication Header in 32-bit words, minus 2.For example, the default length of the authentication data field is 96 bits, or three 32-bit words. With a three-word fixed header, there are a total of six words in the header,and the Payload Length field has a value of 4.
- **Reserved (16 bits):** For future use.
- **Security Parameters Index (32 bits):** Identifies a security association.
- **Sequence Number (32 bits):** A monotonically increasing counter value, discussed later.
- **Authentication Data (variable):** A variable-length field (must be an integral numberof 32-bit words) that contains the Integrity Check Value (ICV), or MAC, for this packet, discussed later.

Anti-Replay Service:

A replay attack is one in which an attacker obtains a copy of an authenticated packet andlater transmits it to the intended destination. The receipt of duplicate, authenticated IPpackets may disrupt service in some way or may have some other undesired consequence. The Sequence Number field is designed to thwart such attacks

When a new SA is established, the **sender** initializes a sequence number counter to 0. Each time that a packet is sent on this SA, the sender increments the counter and places the valuein the Sequence Number field. Thus, the first value to be used is 1. If anti-replay is enabled (the default), the sender must not allow the sequence number to cycle past $2^{32} - 1$ back to zero. Otherwise, there would be multiple valid packets with the same sequence number. Ifthe limit of $2^{32} - 1$ is reached, the sender should terminate this SA and negotiate a new SA with a new key.

Because IP is a connectionless, unreliable service, the protocol does not guarantee that packets will be

delivered in order and does not guarantee that all packets will be delivered. Therefore, the IPSec authentication document dictates that the **receiver** should implement a window of size *W*, with a default of *W* = 64. The right edge of the window represents the highest sequence number, *N*, so far received for a valid packet. For any packet with a sequence number in the range from *N* - *W* + 1 to *N* that has been correctly received (i.e., properly authenticated), the corresponding slot in the window is marked (Figure 1.4). Inbound processing proceeds as follows when a packet is received:

- If the received packet falls within the window and is new, the MAC is checked. If thepacket is authenticated, the corresponding slot in the window is marked.
- If the received packet is to the right of the window and is new, the MAC is checked. If the packet is authenticated, the window is advanced so that this sequence number isthe right edge of the window, and the corresponding slot in the window is marked.
- If the received packet is to the left of the window, or if authentication fails, the packetis discarded; this is an auditable event.

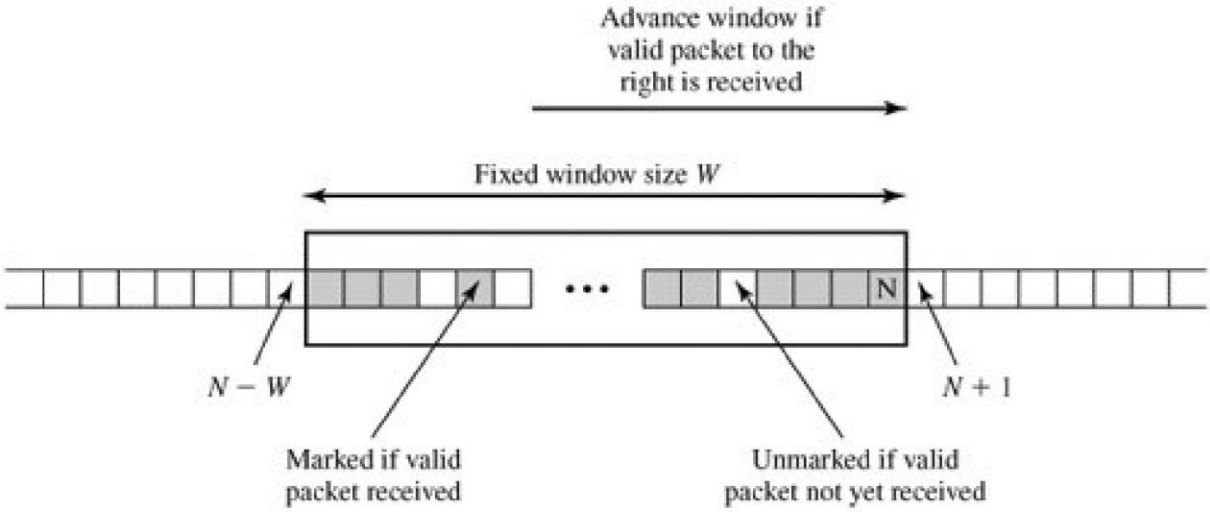


Figure 1.4 Antireplay Mechanism

Integrity Check Value:

The Authentication Data field holds a value referred to as the Integrity Check Value. The ICV is a message authentication code or a truncated version of a code produced by a MAC algorithm. The current specification dictates that a compliant implementation must support

- HMAC-MD5-96
- HMAC-SHA-1-96

Both of these use the HMAC algorithm, the first with the MD5 hash code and the secondwith the SHA-1 hash code. In both cases, the full HMAC value is calculated but then truncated by using the first 96 bits, which is the default length for the Authentication Data field.

The MAC is calculated over

- IP header fields that either do not change in transit (immutable) or that are predictablein value upon arrival at the endpoint for the AH SA. Fields that may change in transit and whose value on arrival is unpredictable are set to zero for purposes of calculation at both source and destination.
- The AH header other than the Authentication Data field. The Authentication Data field is set to zero for purposes of calculation at both source and destination.
- The entire upper-level protocol data, which is assumed to be immutable in transit(e.g., a TCP segment or an inner IP packet in tunnel mode).

For IPv4, examples of immutable fields are Internet Header Length and Source Address. Anexample of a mutable but predictable field is the Destination Address (with loose or strictsource routing). Examples of mutable fields that are zeroed prior to ICV calculation are the Time to Live and Header Checksum fields. Note that both source and destination addressfields are protected, so that address spoofing is prevented.

Transport and Tunnel Modes:

Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH or ESPfields are added to the IP packet, the entire packet plus security fields is treated as thepayload of new "outer" IP packet with a new outer IP header. The entire original, or inner, packet travels through a "tunnel" from one point of an IP network to another; no routersalong the way are able to examine the inner IP header. Because the original packet isencapsulated, the new, larger packet may have totally different source and destinationaddresses, adding to the security. Tunnel mode is used when one or both ends of an SA are asecurity gateway, such as a firewall or router that implements IPSec. With tunnel mode, anumber of hosts on networks behind firewalls may engage in secure communications withoutimplementing IPSec. The unprotected packets generated by such hosts are tunneled throughexternal networks by tunnel mode SAs set up by the IPSec software in the firewall or securerouter at the boundary of the local network.

ESP in tunnel mode encrypts and optionally authenticates the entire inner IP packet, including the inner IP header. AH in tunnel mode authenticates the entire inner IP packet andselected portions of the outer IP header.

Table 1.2 summarizes transport and tunnel mode functionality.

Table 1.2. Tunnel Mode and Transport Mode Functionality

	Transport Mode SA	**Tunnel Mode SA**
AH	Authenticates IP payload and selected portions of IPheader and IPv6 extension headers.	Authenticates entire inner IP packet (inner header plus IP payload) plus selected portions ofouter IP header and outer IPv6 extension headers.

ESP	Encrypts IP payload and any IPv6 extensionheaders following the ESP header.	Encrypts entire inner IP packet.
ESP with Authentication	Encrypts IP payload and any IPv6 extension headers following the ESP header. Authenticates IPpayload but not IP header.	Encrypts entire inner IP packet.Authenticates inner IP packet.

Figure 1.5 shows two ways in which the IPSec authentication service can be used. In one case, authentication is provided directly between a server and client workstations; the workstation can be either on the same network as the server or on an external network. As long as the workstation and the server share a protected secret key, the authentication process is secure. This case uses a transport mode SA. In the other case, a remote workstation authenticates itself to the corporate firewall, either for access to the entire internal network or because the requested server does not support the authentication feature. This case uses a tunnel mode SA.

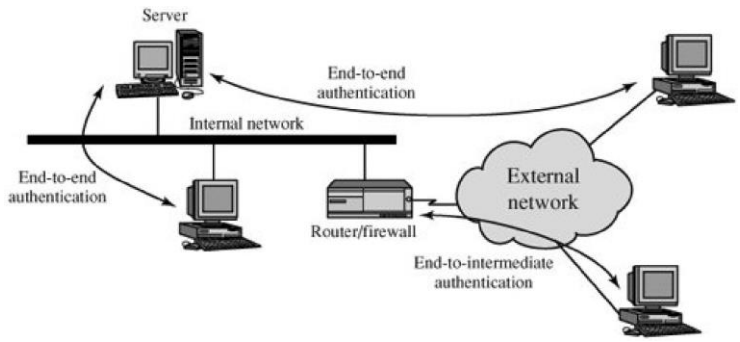


Figure 1.5 End-to-End versus End-to-Intermediate Authentication

Now we look at the scope of authentication provided by AH and the authentication header location for the two modes. The considerations are somewhat different for IPv4 and IPv6. Figure 1.6a shows typical IPv4 and IPv6 packets. In this case, the IP payload is a TCP segment; it could also be a data unit for any other protocol that uses IP, such as UDP or ICMP.

For **transport mode AH** using IPv4, the AH is inserted after the original IP header and before the IP payload (e.g., a TCP segment); this is shown in the upper part of Figure 1.6b. Authentication covers the entire packet, excluding mutable fields in the IPv4 header that are set to zero for MAC calculation.

In the context of IPv6, AH is viewed as an end-to-end payload; that is, it is not examined or processed by intermediate routers. Therefore, the AH appears after the IPv6 base header and the hop-by-hop, routing, and fragment extension headers. The destination options extension header could appear before or after the AH header, depending on the semantics desired. Again, authentication covers the entire packet, excluding mutable fields that are set to zerofor MAC calculation.

For tunnel mode AH, the entire original IP packet is authenticated, and the AH is insertedbetween

the original IP header and a new outer IP header (Figure 1.6c). The inner IP headercarries the ultimate source and destination addresses, while an outer IP header may contain different IP addresses (e.g., addresses of firewalls or other security gateways).

With tunnel mode, the entire inner IP packet, including the entire inner IP header is protectedby AH. The outer IP header (and in the case of IPv6, the outer IP extension headers) is protected except for mutable and unpredictable fields.

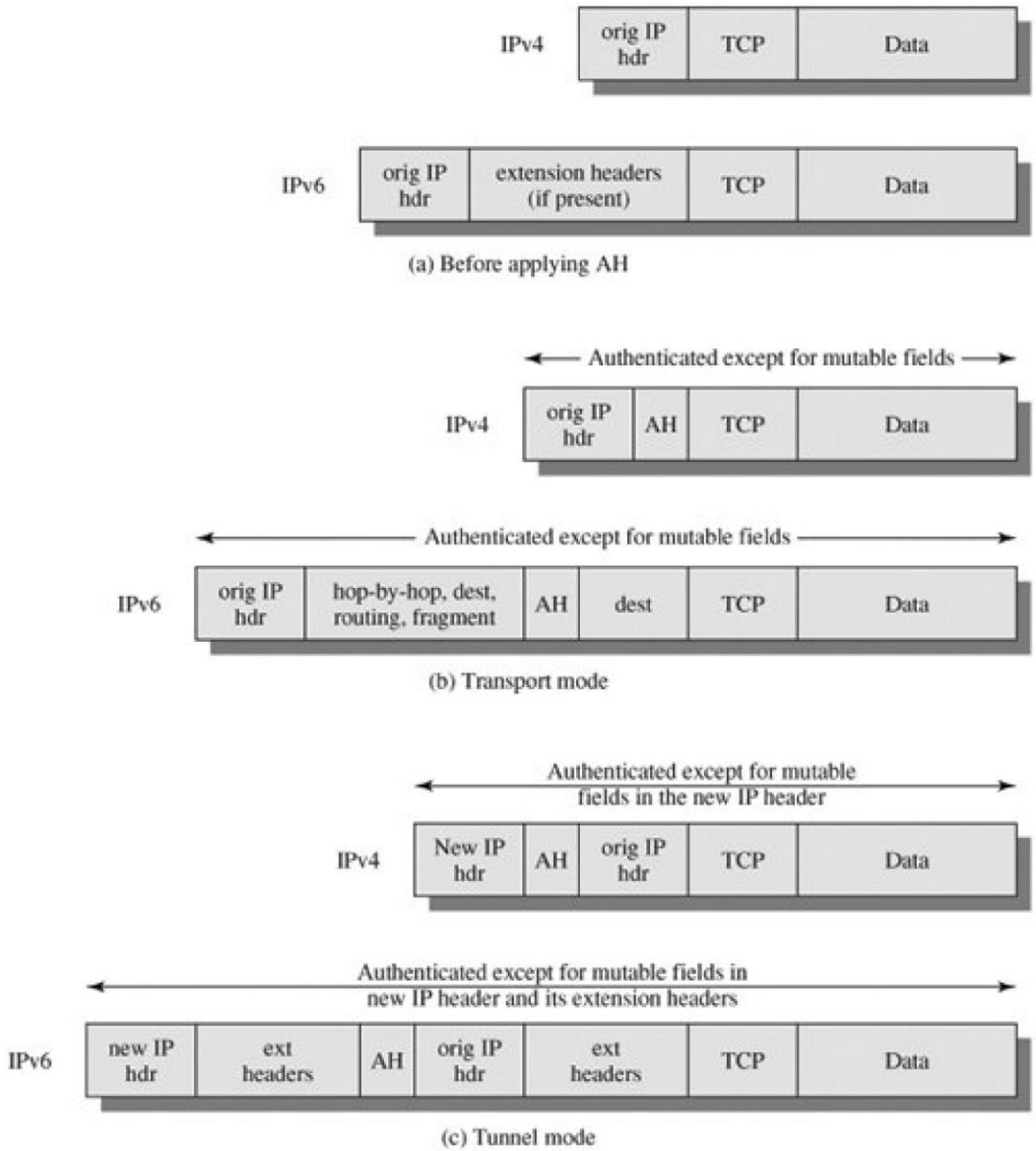


Figure 1.6. Scope of AH Authentication

Encapsulating Security Payload:

The Encapsulating Security Payload provides confidentiality services, including confidentiality of message contents and limited traffic flow confidentiality. As an optional feature, ESP can also provide an authentication service.

ESP Format:

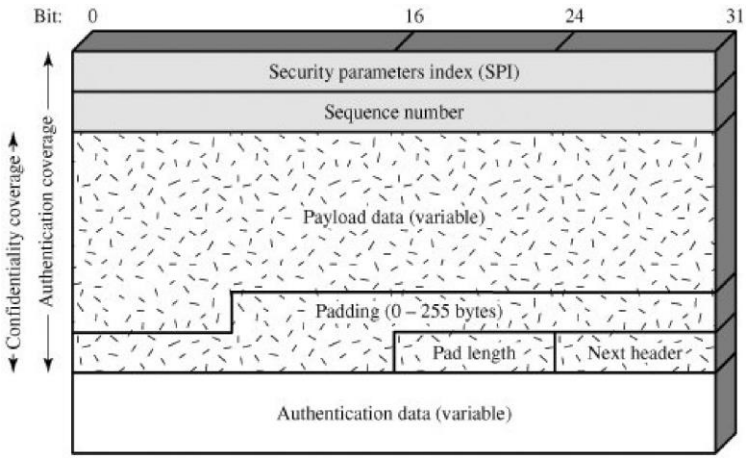


Figure 1.7. IPSec ESP format

Figure 1.7 shows the format of an ESP packet. It contains the following fields:

- Security Parameters Index (32 bits): Identifies a security association.
- Sequence Number (32 bits): A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.
- Payload Data (variable): This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
- Padding (0255 bytes): The purpose of this field is discussed later.
- Pad Length (8 bits): Indicates the number of pad bytes immediately preceding this field.
- Next Header (8 bits): Identifies the type of data contained in the payload data field by identifying the first header in that payload
- Authentication Data (variable): A variable-length field (must be an integral number of32-bit words) that contains the Integrity. Check Value computed over the ESP packet minus the Authentication Data field.

Encryption and Authentication Algorithms:

The Payload Data, Padding, Pad Length, and Next Header fields are encrypted by the ESPservice. If the algorithm used to encrypt the payload requires cryptographic synchronizationdata, such as an initialization vector (IV), then these data may be carried explicitly at the beginning of the Payload Data field. If included, an IV is usually not encrypted, although it isoften referred to as being part of the ciphertext.

The current specification dictates that a compliant implementation must support DES in cipher block

chaining (CBC) mode. A number of other algorithms have been assigned identifiers in the DOI document and could therefore easily be used for encryption; these include

- Three-key triple DES
- RC5
- IDEA
- Three-key triple IDEA
- CAST
- Blowfish

As with AH, ESP supports the use of a MAC with a default length of 96 bits. Also as with AH, the current specification dictates that a compliant implementation must support HMAC- MD5-96 and HMAC-SHA-1-96.

Padding:

The Padding field serves several purposes:

- If an encryption algorithm requires the plaintext to be a multiple of some number of bytes (e.g., the multiple of a single block for a block cipher), the Padding field is usedto expand the plaintext (consisting of the Payload Data, Padding, Pad Length, and Next Header fields) to the required length.
- The ESP format requires that the Pad Length and Next Header fields be right aligned within a 32-bit word. Equivalently, the ciphertext must be an integer multiple of 32 bits. The Padding field is used to assure this alignment.
- Additional padding may be added to provide partial traffic flow confidentiality by concealing the actual length of the payload.

Transport and Tunnel Modes:

Figure 1.8 shows two ways in which the IPSec ESP service can be used. In the upper part of the figure, encryption (and optionally authentication) is provided directly between two hosts. Figure 1.8b shows how tunnel mode operation can be used to set up a *virtual private network*. In this example, an organization has four private networks interconnected across the Internet. Hosts on the internal networks use the Internet for transport of data but do not interact with other Internet-based hosts. By terminating the tunnels at the security gateway to each internal network, the configuration allows the hosts to avoid implementing the security capability. The former technique is support by a transport mode SA, while the latter technique uses a tunnel mode SA.

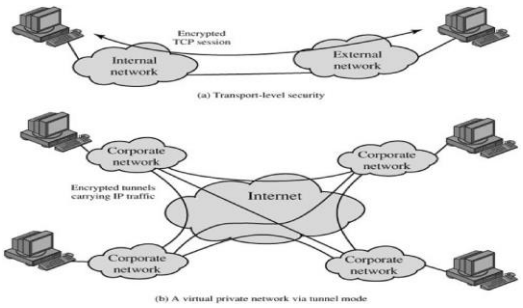
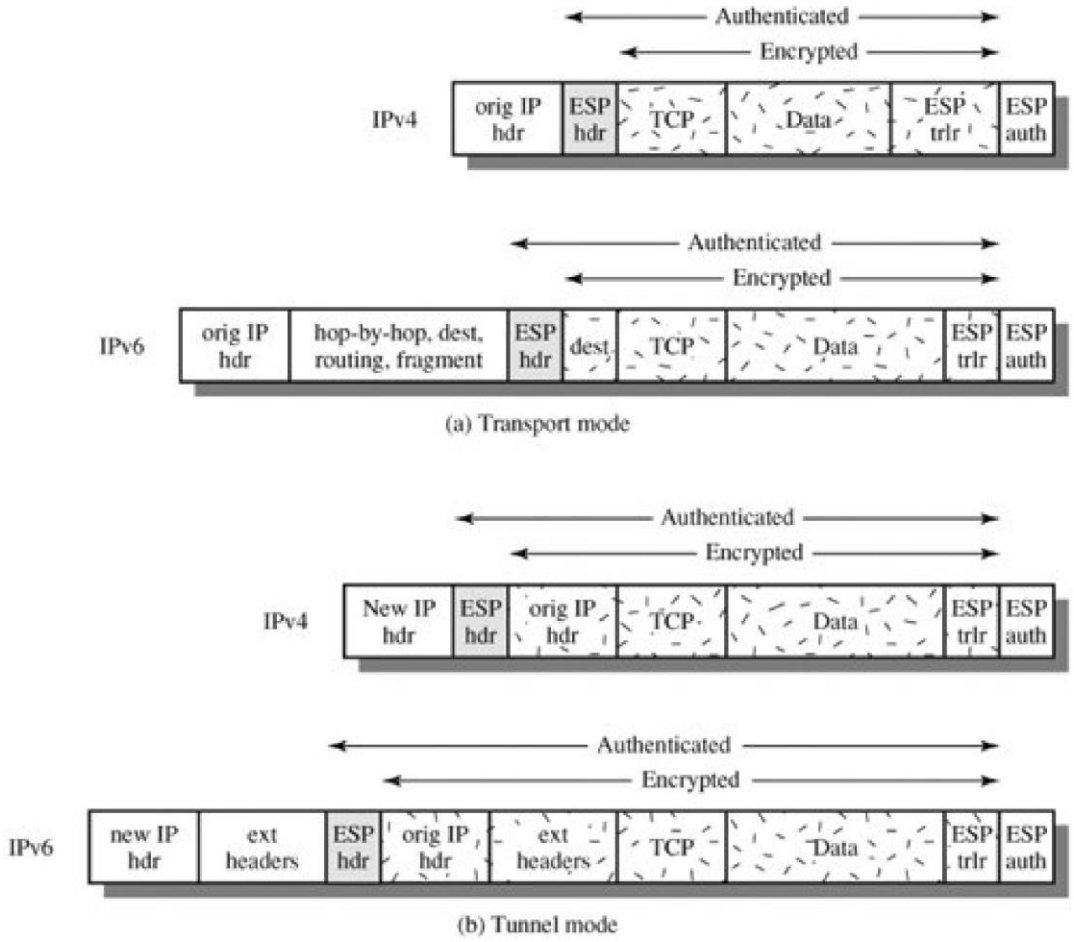


Figure 1.8. Transport-Mode vs. Tunnel-Mode Encryption

Transport Mode ESP:

Transport mode ESP is used to encrypt and optionally authenticate the data carried by IP (e.g., a TCP segment), as shown in Figure 1.9a. For this mode using IPv4, the ESP header is inserted into the IP packet immediately prior to the transport-layer header (e.g., TCP, UDP, ICMP) and an ESP trailer (Padding, Pad Length, and Next Header fields) is placed after the IP packet; if authentication is selected, the ESP Authentication Data field is added after the ESP trailer. The entire transport-level segment plus the ESP trailer are encrypted. Authentication covers all of the ciphertext plus the ESP



header.

Figure 1.9. Scope of ESP Encryption and Authentication

In the context of IPv6, ESP is viewed as an end-to-end payload; that is, it is not examined or processed by intermediate routers. Therefore, the ESP header appears after the IPv6 base header and the hop-by-hop, routing, and fragment extension headers. The destination options extension header could appear before or after the ESP header, depending on the semantics desired. For IPv6, encryption covers the entire transport-level segment plus the ESP trailer plus the destination options extension header if it occurs after the ESP header. Again,authentication covers the ciphertext plus the

ESP header.

Transport mode operation may be summarized as follows:

- At the source, the block of data consisting of the ESP trailer plus the entire transport- layer segment is encrypted and the plaintext of this block is replaced with its ciphertext to form the IP packet for transmission. Authentication is added if thisoption is selected.
- The packet is then routed to the destination. Each intermediate router needs to examine and process the IP header plus any plaintext IP extension headers but does not need to examine the ciphertext.
- The destination node examines and processes the IP header plus any plaintext IP extension headers. Then, on the basis of the SPI in the ESP header, the destination node decrypts the remainder of the packet to recover the plaintext transport-layer segment.

Transport mode operation provides confidentiality for any application that uses it, thus avoiding the need to implement confidentiality in every individual application. This mode of operation is also reasonably efficient, adding little to the total length of the IP packet. One drawback to this mode is that it is possible to do traffic analysis on the transmitted packets.

Tunnel Mode ESP:

Tunnel mode ESP is used to encrypt an entire IP packet (Figure 1.9b). For this mode, theESP header is prefixed to the packet and then the packet plus the ESP trailer is encrypted. This method can be used to counter traffic analysis.

The transport mode is suitable for protecting connections between hosts that support the ESP feature, the tunnel mode is useful in a configuration that includes a firewall or other sort of security gateway that protects a trusted network from external networks. In this latter case, encryption occurs only between an external host and the security gateway or between two security gateways. This relieves hosts on the internal network of the processing burden of encryption and simplifies the key distribution task by reducing the number of needed keys. Further, it thwarts traffic analysis based on ultimate destination.

Consider a case in which an external host wishes to communicate with a host on an internal network protected by a firewall, and in which ESP is implemented in the external host andthe firewalls. The following steps occur for transfer of a transport-layer segment from the external host to the internal host:

- The source prepares an inner IP packet with a destination address of the targetinternal host. This packet is prefixed by an ESP header; then the packet and ESP trailer are encrypted and Authentication Data may be added. The resulting block is encapsulated with a new IP header (base header plus optional extensions such as routing and hop-by-hop options for IPv6)

whose destination address is the firewall; this forms the outer IP packet.

- The outer packet is routed to the destination firewall. Each intermediate router needs to examine and process the outer IP header plus any outer IP extension headers but does not need to examine the ciphertext.
- The destination firewall examines and processes the outer IP header plus any outer IP extension headers. Then, on the basis of the SPI in the ESP header, the destination node decrypts the remainder of the packet to recover the plaintext inner IP packet. This packet is then transmitted in the internal network.
- The inner packet is routed through zero or more routers in the internal network to the destination host.

Combining Security Associations:

An individual SA can implement either the AH or ESP protocol but not both. Sometimes a particular traffic flow will call for the services provided by both AH and ESP. Further, a particular traffic flow may require IPSec services between hosts and, for that same flow, separate services between security gateways, such as firewalls. In all of these cases, multiple SAs must be employed for the same traffic flow to achieve the desired IPSec services. The term *security association bundle* refers to a sequence of SAs through which traffic must be processed to provide a desired set of IPSec services. The SAs in a bundle may terminate at different endpoints or at the same endpoints.

Security associations may be combined into bundles in two ways:

- **Transport adjacency:** Refers to applying more than one security protocol to thesame IP packet, without invoking tunneling. This approach to combining AH and ESP allows for only one level of combination; further nesting yields no added benefit since the processing is performed at one IPsec instance: the (ultimate) destination.
- **Iterated tunneling:** Refers to the application of multiple layers of security protocols effected through IP tunneling. This approach allows for multiple levels of nesting, since each tunnel can originate or terminate at a different IPsec site along the path.

The two approaches can be combined, for example, by having a transport SA between hosts travel part of the way through a tunnel SA between security gateways.

One interesting issue that arises when considering SA bundles is the order in which authentication and encryption may be applied between a given pair of endpoints and the ways of doing so. We examine that issue next. Then we look at combinations of SAs that involveat least one tunnel.

Authentication Plus Confidentiality:

Encryption and authentication can be combined in order to transmit an IP packet that hasboth confidentiality and authentication between hosts. We look at several approaches.

ESP with Authentication Option

This approach is illustrated in Figure 1.9. In this approach, the user first applies ESP to the data to be protected and then appends the authentication data field. There are actually two subcases:

- **Transport mode ESP:** Authentication and encryption apply to the IP payload delivered to the host, but the IP header is not protected.
- **Tunnel mode ESP:** Authentication applies to the entire IP packet delivered to the outer IP destination address (e.g., a firewall), and authentication is performed at that destination. The entire inner IP packet is protected by the privacy mechanism, for delivery to the inner IP destination.

For both cases, authentication applies to the ciphertext rather than the plaintext.

. **Transport Adjacency:**

Another way to apply authentication after encryption is to use two bundled transport SAs,with the inner being an ESP SA and the outer being an AH SA. In this case ESP is usedwithout its authentication option. Because the inner SA is a transport SA, encryption isapplied to the IP payload. The resulting packet consists of an IP header (and possibly IPv6header extensions) followed by an ESP. AH is then applied in transport mode, so thatauthentication covers the ESP plus the original IP header (and extensions) except for mutablefields. The advantage of this approach over simply using a single ESP SA with the ESPauthentication option is that the authentication covers more fields, including the source anddestination IP addresses. The disadvantage is the overhead of two SAs versus one SA. **Transport-Tunnel Bundle:**

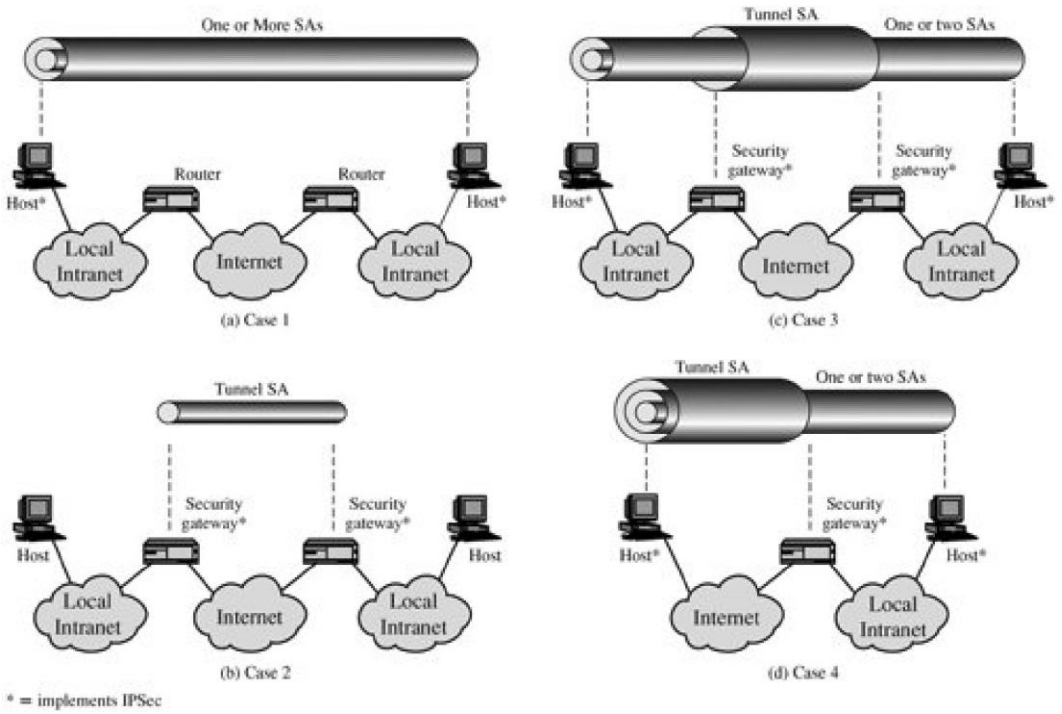
The use of authentication prior to encryption might be preferable for several reasons. First,because the authentication data are protected by encryption, it is impossible for anyone tointercept the message and alter the authentication data without detection. Second, it may be desirable to store the authentication information with the message at the destination for laterreference. It is more convenient to do this if the authentication information applies to theunencrypted message; otherwise the message would have to be reencrypted to verify the authentication information.

One approach to applying authentication before encryption between two hosts is to use abundle consisting of an inner AH transport SA and an outer ESP tunnel SA. In this case,authentication is applied to the IP payload plus the IP header (and extensions) except for mutable fields. The resulting IP packet is then processed in tunnel mode by ESP; the result isthat the entire, authenticated inner packet is encrypted and a new outer IP header (andextensions) is added.

Basic Combinations of Security Associations:

The IPSec Architecture document lists four examples of combinations of SAs that must be supported by compliant IPSec hosts (e.g.workstation, server) or security gateways (e.g. firewall, router). These are illustrated in Figure 1.10. The lower part of each case in the figure represents the physical connectivity of the elements; the upper part represents logical connectivity via one or more nested SAs. Each SA can be either AH or ESP. For host-to-host SAs, the mode may be either transport or tunnel; otherwise it must be tunnel mode.

Figure 1.10 Basic Combinations of Security Associations



In **Case 1,** all security is provided between end systems that implement IPSec. For any two end systems to communicate via an SA, they must share the appropriate secret keys. Amongthe possible combinations:

a. AH in transport mode

b. ESP in transport mode

c. ESP followed by AH in transport mode (an ESP SA inside an AH SA)

d. Any one of a, b, or c inside an AH or ESP in tunnel mode

We have already discussed how these various combinations can be used to support authentication, encryption, authentication before encryption, and authentication after encryption.

For **Case 2,** security is provided only between gateways (routers, firewalls, etc.) and no hostsimplement IPSec. This case illustrates simple virtual private network support. The security architecture document specifies that only a single tunnel SA is needed for this case.

The tunnel could support AH, ESP, or ESP with the authentication option. Nested tunnels arenot required because the IPSec services apply to the entire inner packet.

Case 3 builds on Case 2 by adding end-to-end security. The same combinations discussed forcases 1 and 2 are allowed here. The gateway-to-gateway tunnel provides either authentication or confidentiality or both for all traffic between end systems. When the gateway-to-gateway tunnel is ESP, it also provides a limited form of traffic confidentiality. Individual hosts can implement any additional. IPSec services required for given applications or given users by means of end-to-end SAs.

Case 4 provides support for a remote host that uses the Internet to reach an organization's firewall and then to gain access to some server or workstation behind the firewall. Onlytunnel mode is required between the remote host and the firewall. As in Case 1, one or two SAs may be used between the remote host and the local host.