KARPAGAM ACADEMY OF HIGHER EDUCATION
(Deemed to be University)
Established Under Section 3 of UGC Act 1956)
Coimbatore - 641021.
(For the candidates admitted from 2018 onwards)
DEPARTMENT OF COMPUTER SCIECE

Semester-I

18CSP102 CRYPTOGRAPHY AND NETWORK SECURITY 4H – 4C

COURSE OBJECTIVE

This course will provide students with a theoretical knowledge to understand the fundamental principles of access control models and techniques and,

- To know about various encryption techniques.
- To understand the concept of Public key cryptography.
- To study about message authentication and hash functions
- To impart knowledge on Network security

COURSE OUTCOME

On successful completion of the course the student should be able to:

- Classify the symmetric encryption techniques
- Illustrate various Public key cryptographic techniques
- Evaluate the authentication and hash algorithms.
- Summarize the intrusion detection and its solutions to overcome the attacks.
- Understand basic concepts of system level security

UNIT I

Introduction – Security Trends - The OSI Security Architecture – Security Attacks – Security Services – Security Mechanisms – A Model for Network Security. Classical Encryption Techniques – Symmetric Cipher Model – Substitution Techniques - Transposition Techniques – Rotor Machines - Steganography.

UNIT -II

Block Ciphers and Data Encryption Standard –Block Cipher Principles – The Data Encryption Standard - The Strength of DES –Advanced Encryption Standard (AES) – Evaluation Criteria for AES – The AES Cipher – Multiple Encryption and Triple DES – Block Cipher Modes of Operation – Stream Ciphers and RC4- modular Arithmetic and Euclidean Algorithm.

UNIT-III

Confidentiality using Symmetric Encryption – Placement of Encryption Function – Traffic Confidentiality – Key Distribution – Public key Cryptography and RSA – Principles of Public Key Cryptosystems – The RSA Algorithm- Basic prime numbers and Discrete Logarithms -Key Management – Diffie Hellman Key Exchange.

UNIT-IV

Message Authentication and hash functions – Authentication Functions – Message Authentication Codes (MAC's) Functions – Security of Hash Functions and MAC's Digital Signatures and Authentication Protocols – Digital Signatures – Digital Signature Standard

UNIT-V

Network Security Applications - Authentication Applications – KERBEROS – X.509 Authentication Service – Public Key Infrastructure – Electronic Mail Security – Pretty Good Privacy – S/MIME – IP Security.

SUGGESTED READINGS

1. William Stallings. 2006. Cryptography and Network Security Principles and Practices(4th ed.). New Delhi: Pearson Education.
   (Page Nos. : 6-35 62-75 80-135 199-220 289-298 317-340 377-390 400-436 436-457 483-506)
2. AtulKahate. 2003. Cryptography and Network Security (2nd ed.). Tata McGraw Hill New Delhi.

3. AnkitFadia. (19980. Network Security(1st ed.). New Delhi: McMillan Publications.
4. Bruce Schneir. (1998). Applied Cryptography (1st ed.). New Delhi: CRC Press.
5. Charlie Kaufman, Radia Perlman, & Mike Speciner. (2003). Network Security Private Communication in a Public World (2nd ed.). New Delhi: Prentice-Hall of India.
6. Menezes, A. Van Oorschot, & Vanstone, S. (1997). Hand Book of Applied Cryptography (1st ed.). New Delhi: CRC Press. (Free Downloadable)

WEB SITES

1. williamstallings.com/Crypto3e.html
2. u.cs.biu.ac.il/~herzbea/book.html
3. cryptofundamentals.com/algorithms

KARPAGAM ACADEMY OF HIGHER EDUCATION

Coimbatore - 641021.

(For the candidates admitted from 2018 onwards)

DEPARTMENT OF COMPUTER SCIENCE, CA & IT

LECTURE PLAN

STAFF NAME: Dr.S.Hemalatha

SUBJECT NAME:CRYPTOGRAPHY AND NETWORK SECURITY

SUBJECT CODE:18CSP102 **SEMESTER: I**

S.No	Lecture Duration	TOPICS TO BE COVERED	Support Materials/Page No's
		UNIT I	
1	1	Introduction to information security,Security Trends ,The OSI security Architecture	T1:6-12
2	1	Security Attacks, Security Services ,Security Mechanisms	T1:12-20
3	1	A model for Network Security	T1:22-23
4	1	Classical Encryption techniques -Introduction	T1: 29
5	1	Symmetric cipher model	T1:30-35, W1
6	1	Substitution techniques (Caesar Cipher,monoalphabetic ciphers, playfair cipher)	T1:35-39
7	1	Substitution techniques Cont.. (Hill cipher, Polyalphabetic cipher)	T1:40-49
8	1	Transposition techniques, Rotor Machines, Steganography	T1:49-54
9	1	Recapitulation and discussion of important question's	
Total no of hours planned for unit I: 9			

UNIT-2

1	1	Block Ciphers and Data encryption Standard, Block Cipher principles	T1:63-72
2	1	The Data Encryption Standard ,Strength of DES	T1:72-83
3	1	Advanced Encryption standard(AES) Evaluation Criteria For AES,	T1:135-141
4	1	The AES Cipher	T1:142-145
5	1	Multiple Encryption	T1:175
6	1	Triple Des , Block Cipher modes of Operation	T1:176-189
7	1	Stream Ciphers And RC4	T1:189-194
8	1	modular Arithmetic and Euclidean Algorithm	T1: 101-109
9	1	Recapitulation and discussion of important question's	
Total no of hours planned for unit II : 9			

UNIT-3

1	1	Confidentiality using symmetric Encryption	T1:200-201
2	1	Placement of encryption function, Traffic confidentiality	T1:201-211
3	1	Key distribution	T1:211-218
4	1	Public-Key Encryption, Principles of Public-key Cryptosystems	T1:257-268
5	1	The RSA Algorithm	T1:268-280, W2
6	1	Basic prime numbers and Discrete Logarithms	T1: 236, 247
7	1	Key management : Public key distribution	T1:290-297
8	1	Diffie-Hellman Key Exchange	T1:298-301, W3
9	1	Recapitulation and discussion of important question's	
Total no of hours planned for unit III : 9			

UNIT-4

1	1	Message Authentication and Hash Function	T1:317,334
2	1	Authentication Function : Message encryption	T1:320-325
3	1	Authentication Function : MAC & Hash	T1:326-330
4	1	Message Authentication Codes(MAC's)Function	T1:331-333
5	1	Security of Hash functions and MAC's	T1:340-343
6	1	Digital signature and Authentication Protocols - Digital signatures	T1:377 -378
7	1	Digital Signature Standard	T1:390-394 R2: 412-422
8	1	Digital Signature Standard Functions	T1:395-397
9	1	Recapitulation and discussion of important question's	
Total no of hours planned for unit IV :9			

UNIT-5

1	1	Network security Applications	T1:398
2	1	Authentication Applications	T1:400-401
3	1	KERBEROS	T1:402-419
4	1	X.509 Authentication service	T1:419-428
5	1	Public Key Infrastructure, Electronic Mail Security	T1:428-436
6	1	Pretty Good privacy	T1:438-457
7	1	s/MiME	T1:457-474
8	1	IP Security	T1:483, R1:345-357
9	1	Recapitulation and discussion of important question's	
10	1	End Semester Question paper Discussion	
11	1	End Semester Question paper Discussion	
12	1	End Semester Question paper Discussion	
Total no of hours planned for unit V : 12			
Total hours Planned: 48			

Text Books

1. William Stallings. 2006. Cryptography and Network Security Principles and Practices(4th ed.). New Delhi: Pearson Education.
2. AtulKahate. 2003. Cryptography and Network Security (2nd ed.). Tata McGraw Hill New Delhi.
3. Charlie Kaufman, Radia Perlman, & Mike Speciner. (2003). Network Security Private Communication in a Public World (2nd ed.). New Delhi: Prentice-Hall of India.
4. Menezes, A. Van Oorschot, & Vanstone, S. (1997). Hand Book of Applied Cryptography (1st ed.). New Delhi: CRC Press. (Free Downloadable)

REFERENCES BOOKS

1. Ankit Fadia. (1998). Network Security (1st ed.). New Delhi: McMillan Publications.

2. Bruce Schneir. (1998). Applied Cryptography (1st ed.). New Delhi: CRC Press.

WEB SITES

1. williamstallings.com/Crypto3e.html
2. u.cs.biu.ac.il/~herzbea/book.html
3. cryptofundamentals.com/algorithms

UNIT I

Syllabus

Introduction – Security Trends - The OSI Security Architecture – Security Attacks – Security Services – Security Mechanisms – A Model for Network Security. Classical Encryption Techniques – Symmetric Cipher Model – Substitution Techniques - Transposition Techniques – Rotor Machines - Steganography.

SECURITY TRENDS

Computer data often travels from one computer to another, leaving the safety of its protected physical surroundings. Once the data is out of hand, people with bad intention could modify or forge your data, either for amusement or for their own benefit. Cryptography can reformat and transform our data, making it safer on its trip between computers. The technology is based on the essentials of secret codes, augmented by modern mathematics that protects our data in powerful ways.

Cryptology:

This is the study of techniques for ensuring the secrecy and/or authenticity of information. The two main branches of cryptology are

Cryptography

It is the study of the design of such techniques

Cryptography, is the practice and study of hiding information.

When a message is sent using cryptography, it is changed (or encrypted) before it is sent. The method of changing text is called a "code" or, more precisely, a "cipher". The changed text is called "ciphertext". The change makes the message hard to read. Someone who wants to read it must change it back (or decrypt it). How to change it back is a secret. Both the person that sends the message and the one that gets it should know the secret way to change it, but other people should not be able to.

Cryptanalysis

It deals with the defeating such techniques, to recover information, or forging information that will be accepted as authentic.

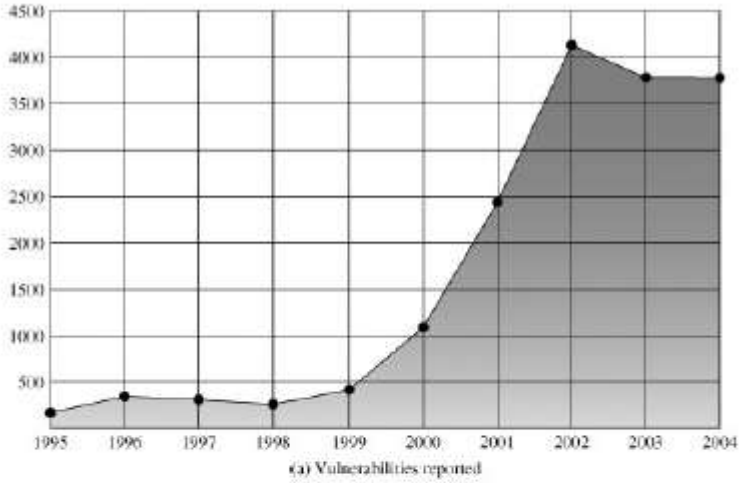
• **Computer Security** - generic name for the collection of tools designed to protect data and to prevent hackers

• **Network Security** - measures to protect data during their transmission

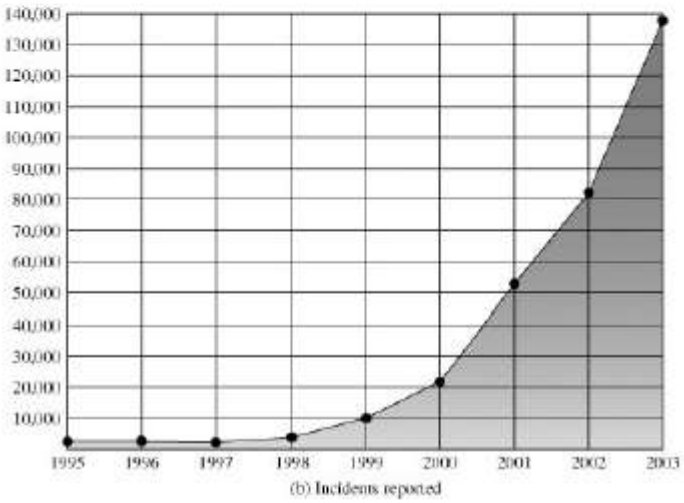
• **Internet Security** - measures to protect data during their transmission over a collection of interconnected networks

Security Trends

In 1994, the Internet Architecture Board (IAB) issued a report entitled "Security in the Internet Architecture" (RFC 1636). The report stated the general consensus that the Internet needs more and better security, and it identified key areas for security mechanisms. Among these were the need to secure the network infrastructure from unauthorized monitoring and control of network traffic and the need to secure end-user-to-end-user traffic using authentication and encryption mechanisms. These concerns are fully justified. As confirmation, consider the trends reported by the Computer Emergency Response Team (CERT) Coordination Center (CERT/CC).



(a) Vulnerabilities reported



(b) Incidents reported

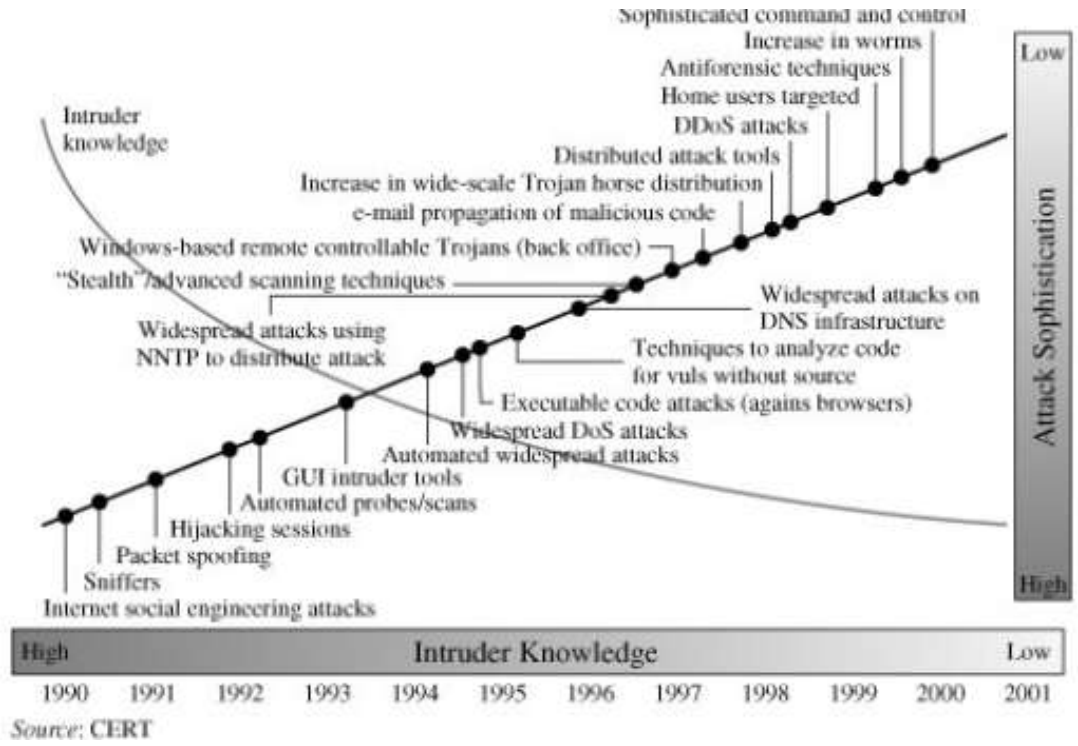


Fig 1.1 Trends in Security

Figure 1.1 shows the trend in Internet-related vulnerabilities reported to CERT over a 10-year period. These include security weaknesses in the operating systems of attached computers (e.g., Windows, Linux) as well as vulnerabilities in Internet routers and other network devices.

Figure 1.1 shows the number of security-related incidents reported to CERT. These include denial of service attacks; IP spoofing, in which intruders create packets with false IP addresses and exploit applications that use authentication based on IP; and various forms of eavesdropping and packet sniffing, in which attackers read transmitted information, including logon information and database contents.

This increase in attacks coincides with an increased use of the Internet and with increases in the complexity of protocols, applications, and the Internet itself. Critical infrastructures increasingly rely on the Internet for operations. Individual users rely on the security of the Internet, email, the Web, and Web-based applications to a greater extent than ever. Thus, a wide range of technologies and tools are needed to counter the growing threat. At a basic level, cryptographic algorithms for confidentiality and authentication assume greater importance. As well, designers need to focus on

Internet-based protocols and the vulnerabilities of attached operating systems and applications. This book surveys all of these technical areas.

OSI SECURITY ARCHITECTURE

To assess effectively the security needs of an organization and to evaluate and choose various security products and policies, the manager responsible for security needs some systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. The OSI security architecture was developed in the context of the OSI protocol architecture. The OSI security architecture provides a useful, if abstract, overview of many of the concepts. The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly in Table 1:

Threat
A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit vulnerability.
Attack
An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.
Table 1: Threats and Attacks

Security Attacks, Services And Mechanisms

To assess the security needs of an organization effectively, the manager responsible for security needs some systematic way of defining the requirements for security and characterization of approaches to satisfy those requirements. One approach is to consider three aspects of information security:

 Security attack – Any action that compromises the security of information owned by an organization.

 Security mechanism – A mechanism that is designed to detect, prevent or recover from a security attack.

 Security service – A service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks and they make use of one or more security mechanisms to provide the service.

SECURITY SERVICES

The classification of security services are as follows:

 Confidentiality: Ensures that the information in a computer system and transmitted information are accessible only for reading by authorized parties.

Eg., printing, displaying and other forms of disclosure.

 Authentication: Ensures that the origin of a message or electronic document is correctly identified, with an assurance that the identity is not false.

Integrity: Ensures that only authorized parties are able to modify computer system assets and transmitted information. Modification includes writing, changing status, deleting, creating and delaying or replaying of transmitted messages.

 Non repudiation: Requires that neither the sender nor the receiver of a message be able to deny the transmission.

 Access control: Requires that access to information resources may be controlled by or the target system.

 Availability: Requires that computer system assets be available to authorized parties when needed.

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.

Table 1.1 Security Services (X.800)

SECURITY MECHANISMS

One of the most specific security mechanisms in use is cryptographic techniques. Encryption or encryption-like transformations of information are the most common means of providing security. Some of the mechanisms are,

i. **SPECIFIC SECURITY MECHANISMS**

- **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.

ii. **PERVASIVE SECURITY MECHANISMS**

- **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.

SECURITY ATTACKS

A useful categorization of these attacks is in terms of

 Passive attacks

 Active attacks

Passive attack 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. Passive attacks are of two types:

Release of message contents: A telephone conversation, an e-mail message and a transferred file may contain sensitive or confidential information. We would like to prevent the opponent from learning the contents of these transmissions.

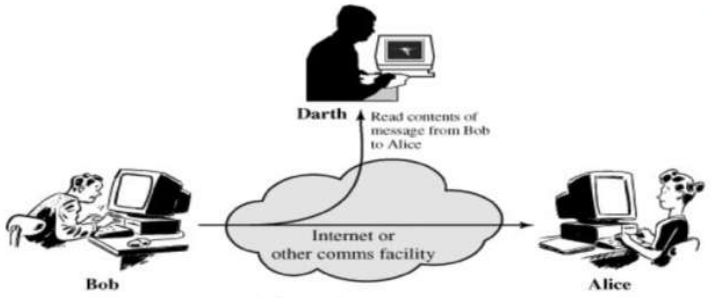


Figure 1.2 Release of message content

 Traffic analysis: If we had encryption protection in place, an opponent might still be able to observe the pattern of the message. The opponent could determine the location and identity of communication hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of communication that was taking place.

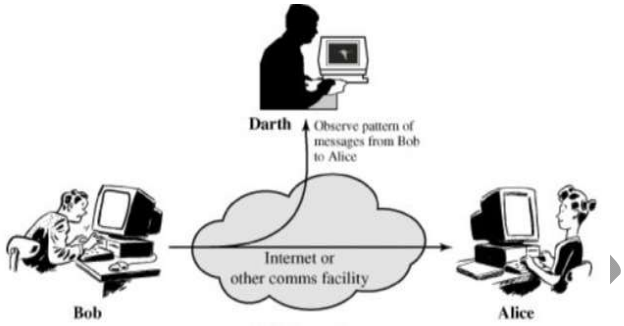


Figure 1.3 Traffic Analysis

Passive attacks are very difficult to detect because they do not involve any alteration of data. However, it is feasible to prevent the success of these attacks.

Active attacks

These attacks involve some modification of the data stream or the creation of a false stream. These attacks can be classified in to four categories:

 Masquerade – One entity pretends to be a different entity.

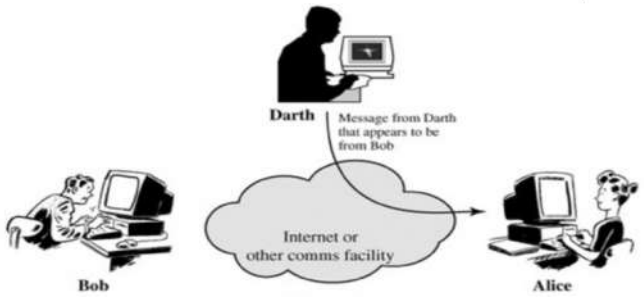


Figure 1.4 Masquerade

 Replay – involves passive capture of a data unit and its subsequent transmission to produce an unauthorized effect.

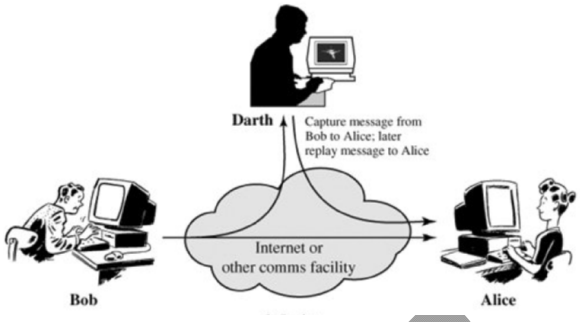


Figure 1.5 Replay

 Modification of messages – Some portion of legitimate message is altered or the messages are delayed or recorded, to produce an unauthorized effect.

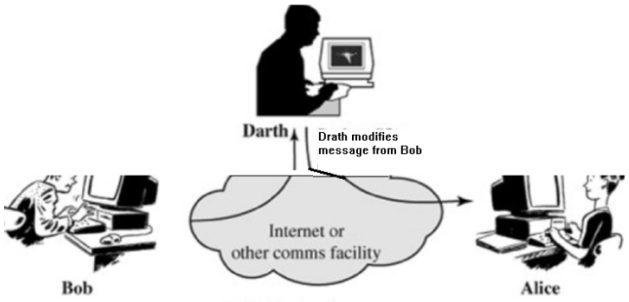


Figure 1.6 Modification of message

 Denial of service – Prevents or inhibits the normal use or management of communication facilities. Another form of service denial is the disruption of an entire network, either by disabling the network or overloading it with messages so as to degrade performance.

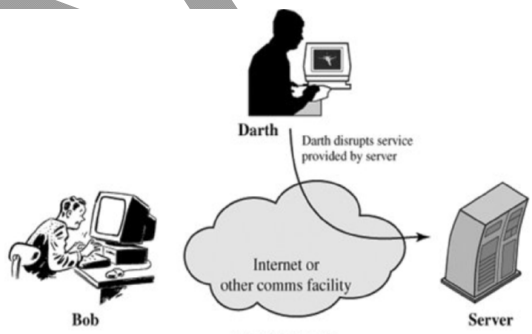


Figure 1.7 Denial of service

It is quite difficult to prevent active attacks absolutely, because to do so would require physical protection of all communication facilities and paths at all times. Instead, the goal is to detect them and to recover from any disruption or delays caused by them.

Symmetric and Asymmetric Encipherment

Encryption/Decryption methods fall into two categories.

- **Symmetric-key Encipherment**
- **Asymmetric Encipherment**

Symmetric-key Encipherment

Symmetric-key encipherment uses a single secret key for both encryption and decryption. Encryption/decryption can be thought of as electronic locking system. In symmetric-key enciphering, Alice puts the message in a box and locks the box using the shared secret key; Bob unlocks the box with the same key and takes out the messages.

In symmetric key algorithms, the encryption and decryption keys are known both to sender and receiver. The encryption key is shared and the decryption key is easily calculated from it. In many cases, the encryption and decryption keys are the same.

Asymmetric Encipherment

In asymmetric encipherment, we have the same situation aas the symmetric-key encipherment, with a few exceptions. First, there are two keys instead of one; one public key and one private key. To send a secure message to Bob, Alice firsts encrypts the message using Bob□s public key.

To decrypts the message, Bob uses his own private key.

In public key cryptography, encryption key is made public, but it is computationally infeasible to find the decryption key without the information known to the receiver.

A MODEL FOR NETWORK SECURITY

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.

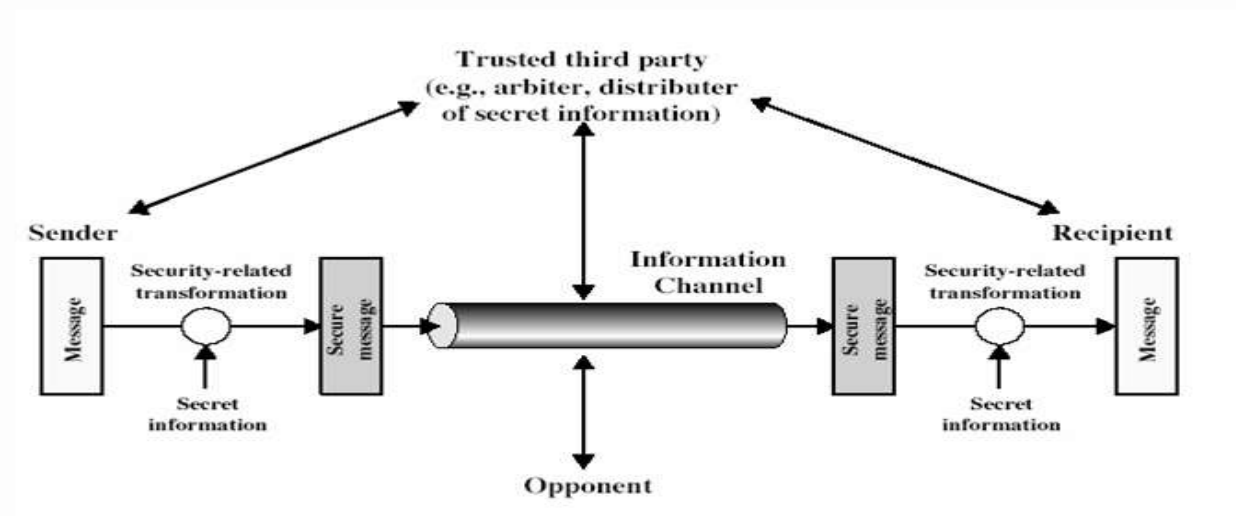


Figure 1.8 Model for Network Security

Using this model requires us to:

– design a suitable algorithm for the security transformation

– generate the secret information (keys) used by the algorithm

– develop methods to distribute and share the secret information

– specify a protocol enabling the principals to use the transformation and secret information for a security service

MODEL FOR NETWORK ACCESS SECURITY

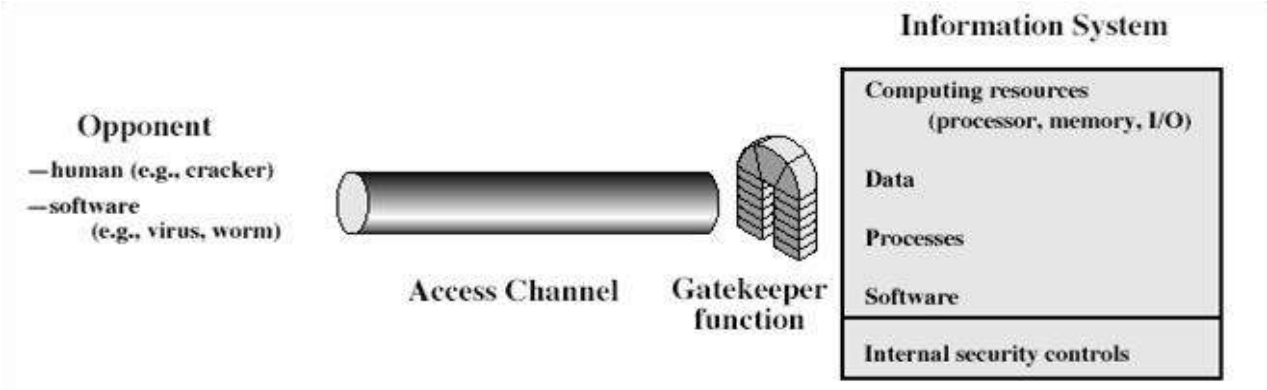


Figure 1.9 Network Access Security Model

• **using this model requires us to:**

– select appropriate gatekeeper functions to identify users

– implement security controls to ensure only authorised users access designated information or resources

• **trusted computer systems can be used to implement this model**

CONVENTIONAL ENCRYPTION

This is also referred conventional / private-key / single-key. Here the sender and recipient share a common key. All classical encryption algorithms are private-key. It was only type prior to invention of public-key in 1970.

Some basic terminologies used :

• **plaintext**- the original message

• **ciphertext**- the coded message

• **cipher**- algorithm for transforming plaintext to ciphertext

• **key**- info used in cipher known only to sender/receiver

• **encipher (encrypt)** - converting plaintext to ciphertext

• **decipher (decrypt)** - recovering ciphertext from plaintext

• **cryptography**- study of encryption principles/methods

• **cryptanalysis (codebreaking)** - the study of principles/ methods of deciphering ciphertext*without*knowing key

• **cryptology**- the field of both cryptography and cryptanalysis

Figure 1.13 shows the simplified model of conventional encryption. Here the original message, referred to as plaintext, is converted into apparently random nonsense, referred to as cipher text. The encryption process consists of an algorithm and a key. The key is a value independent of the plaintext. Changing the key changes the output of the algorithm. Once the cipher text is produced, it may be transmitted. Upon reception, the cipher text can be transformed back to the original plaintext by using a decryption algorithm and the same key that was used for encryption. The security depends on several factors. First, the encryption algorithm must be powerful enough that it is impractical to decrypt a message on the basis of cipher text alone. Beyond that, the security depends on the secrecy of the key, not the secrecy of the algorithm.

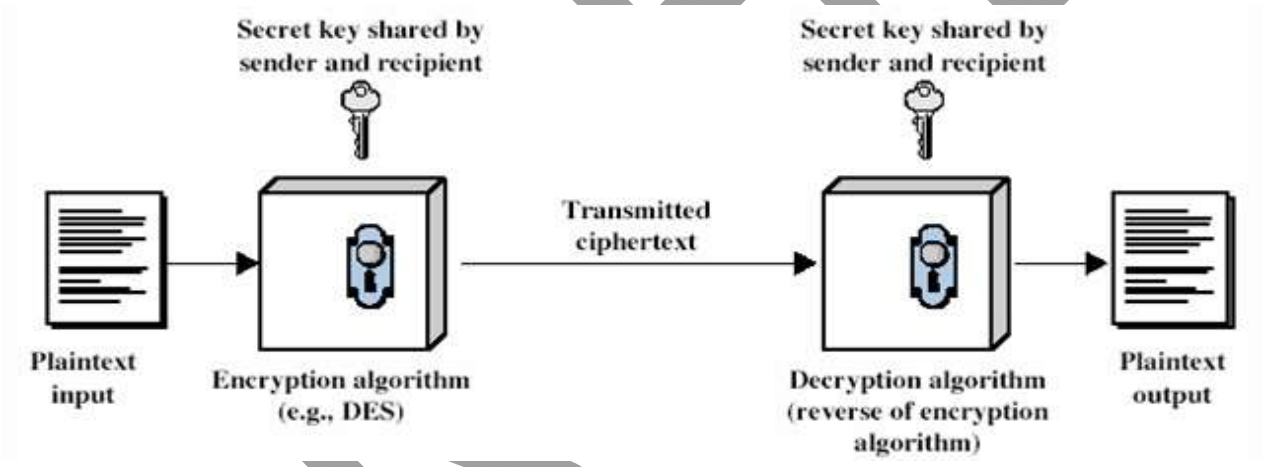


Figure 1.10 Simplified Model of Conventional Encryption

• **Two requirements for secure use of symmetric encryption:**

– a strong encryption algorithm

– a secret key known only to sender / receiver

$Y = EK(X)$

$X = DK(Y)$

• **assume encryption algorithm is known**

• **implies a secure channel to distribute key**

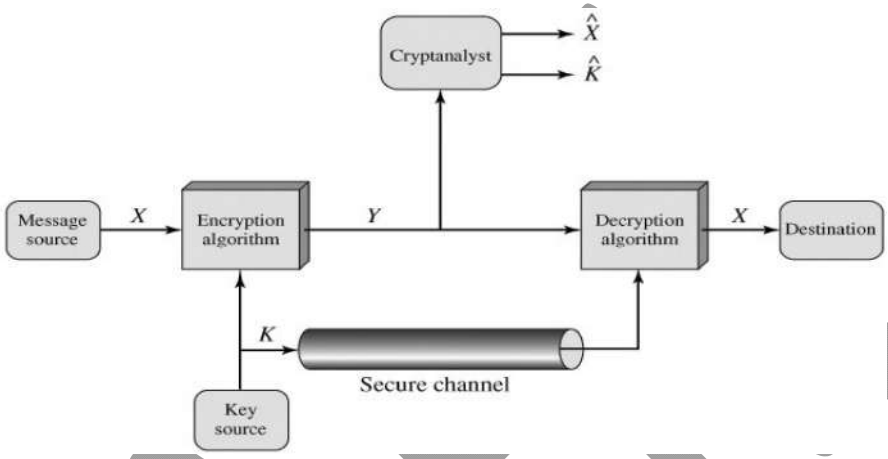


Figure 1.11 Model of Conventional Cryptosystem

A source produces a message in plaintext, X = [X1, X2, … , XM] where M are the number of letters in the message. A key of the form K = [K1, K2, …, KJ] is generated. If the key is generated at the source, then it must be provided to the destination by means of some secure channel. With the message X and the encryption key K as input, the encryption algorithm forms the cipher text Y = [Y1, Y2, …, YN]. This can be expressed as Y = EK(X) The intended receiver, in possession of the key, is able to invert the transformation: X = DK(Y) An opponent, observing Y but not having access to K or X, may attempt to recover X or K or both. It is assumed that the opponent knows the encryption and decryption algorithms. If the opponent is interested in only this particular message, then the focus of effort is to recover X by generating a plaintext estimate. Often if 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.

CRYPTOGRAPHY

Cryptographic systems are generally classified along 3 independent dimensions:

 Type of operations used for transforming plain text to cipher text

All the encryption algorithms are abased on two general principles: substitution, in which each element in the plaintext is mapped into another element, and transposition, in which elements in the plaintext are rearranged.

 The number of keys used

If the sender and receiver uses same key then it is said to be symmetric key (or) single key (or) conventional encryption. If the sender and receiver use different keys then it is said to be public key encryption.

 The way in which the plain text is processed

A block cipher processes the input and block of elements at a time, producing output block for each input block. A stream cipher processes the input elements continuously, producing output element one at a time, as it goes along.

CRYPTANALYSIS

The process of attempting to discover X or K or both is known as cryptanalysis. The strategy used by the cryptanalysis depends on the nature of the encryption scheme and the information available to the cryptanalyst.

There are various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst.

 Cipher text only – A copy of cipher text alone is known to the cryptanalyst.

 Known plaintext – The cryptanalyst has a copy of the cipher text and the corresponding plaintext.

 Chosen plaintext – The cryptanalysts gains temporary access to the encryption machine. They cannot open it to find the key, however; they can encrypt a large number of suitably chosen plaintexts and try to use the resulting cipher texts to deduce the key.

. **Chosen cipher text** – The cryptanalyst obtains temporary access to the decryption machine, uses it to decrypt several string of symbols, and tries to use the results to deduce the key.

CLASSICAL ENCRYPTION TECHNIQUES

There are two basic building blocks of all encryption techniques: substitution and transposition.

1.2.1 .SUBSTITUTION TECHNIQUES

A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols. If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with cipher text bit patterns.

(i)**Caesar cipher (or) shift cipher**

It is a mono-alphabetic cipher wherein each letter of the plaintext is substituted by another letter to form the ciphertext. It is a simplest form of substitution cipher scheme.

This cryptosystem is generally referred to as the **Shift Cipher**. The concept is to replace each alphabet by another alphabet which is 'shifted' by some fixed number between 0 and 25.

For this type of scheme, both sender and receiver agree on a 'secret shift number' for shifting the alphabet. This number which is between 0 and 25 becomes the key of encryption.

The name 'Caesar Cipher' is occasionally used to describe the Shift Cipher when the 'shift of three' is used.

Plaintext Alphabet	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
Ciphertext Alphabet	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

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 3 places further down the alphabet.

e.g., plain text : pay more money

Cipher text: SDB PRUH PRQHB

Note that the alphabet is wrapped around, so that letter following „z□ is „a□. For each plaintext letter p, substitute the cipher text letter c such that

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

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

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

where k takes on a value in the range 1 to 25.

The decryption algorithm is simply

$$P = D(C) = (C-k) \bmod 26$$

Monoalphabetic Cipher

Monoalphabetic cipher is a substitution cipher in which for a given key, the cipher alphabet for each plain alphabet is fixed throughout the encryption process. For example, if 'A' is encrypted as 'D', for any number of occurrence in that plaintext, 'A' will always get encrypted to 'D'.

It is an improvement to the Caesar Cipher. Instead of shifting the alphabets by some number, this scheme uses some permutation of the letters in alphabet.

For example, A.B…..Y.Z and Z.Y……B.A are two obvious permutation of all the letters in

alphabet. Permutation is nothing but a jumbled up set of alphabets.

With 26 letters in alphabet, the possible permutations are 26! (Factorial of 26) which is equal to $4x10^{26}$. The sender and the receiver may choose any one of these possible permutation as a ciphertext alphabet. This permutation is the secret key of the scheme.

Process of Simple Substitution Cipher

- Write the alphabets A, B, C,...,Z in the natural order.
- The sender and the receiver decide on a randomly selected permutation of the letters of the alphabet.
- Underneath the natural order alphabets, write out the chosen permutation of the letters of the alphabet. For encryption, sender replaces each plaintext letters by substituting the permutation letter that is directly beneath it in the table. This process is shown in the following illustration. In this example, the chosen permutation is K,D, G, ..., O. The plaintext 'point' is encrypted to 'MJBXZ'.

Here is a jumbled Ciphertext alphabet, where the order of the ciphertext letters is a key.

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

- On receiving the ciphertext, the receiver, who also knows the randomly chosen permutation, replaces each ciphertext letter on the bottom row with the corresponding plaintext letter in the top row. The ciphertext 'MJBXZ' is decrypted to 'point'.

Polyalphabetic ciphers

Polyalphabetic Cipher is a substitution cipher in which the cipher alphabet for the plain alphabet may be different at different places during the encryption process. The next two examples, **playfair and Vigenere Cipher are 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 cipher. All the techniques have the following features in common.
- □ A set of related monoalphabetic substitution rules are used
- □ A key determines which particular rule is chosen for a given transformation.

Playfair Cipher

In this scheme, pairs of letters are encrypted, instead of single letters as in the case of simple substitution cipher.

In playfair cipher, initially a key table is created. The key table is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table as we need only 25 alphabets instead of 26. If the plaintext contains J, then it is replaced by I.

The sender and the receiver deicide on a particular key, say 'tutorials'. In a key table, the first characters (going left to right) in the table is the phrase, excluding the duplicate letters. The rest of the table will be filled with the remaining letters of the alphabet, in natural order. The key table works out to be −

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

Process of Playfair Cipher

- First, a plaintext message is split into pairs of two letters (digraphs). If there is an odd number of letters, a Z is added to the last letter. Let us say we want to encrypt the message "hide money". It will be written as −

  HI DE MO NE YZ

- The rules of encryption are −

  - If both the letters are in the same column, take the letter below each one (going back to the top if at the bottom)

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

'H' and 'I' are in same column, hence take letter below them to replace. HI → QC

- If both letters are in the same row, take the letter to the right of each one (going back to the left if at the farthest right)

T	U	O	R	I
A	L	S	B	C

'D' and 'E' are in same row, hence take letter to the right of them to replace. DE → EF

D	E	F	G	H
K	M	N	P	Q
V	W	X	Y	Z

- If neither of the preceding two rules are true, form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

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

'M' and 'O' nor on same column or same row, hence form rectangle as shown, and replace letter by picking up opposite corner letter on same row
MO -> NU

Using these rules, the result of the encryption of 'hide money' with the key of 'tutorials' would be −

QC EF NU MF ZV

Decrypting the Playfair cipher is as simple as doing the same process in reverse. Receiver has the same key and can create the same key table, and then decrypt any messages made using that key.

Example 2: (Playfair cipher)

The playfair algorithm is based on the use of 5x5 matrix of letters constructed using a keyword. Let **the keyword be "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 alphabetical order.The letter "i□ and "j□ count as one letter. Plaintext is encrypted two letters at a time according to the following rules:

□ Repeating plaintext letters that would fall in the same pair are separated with a filler letter such as "x□.

□ 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 following the last.

□ Plaintext letters that fall in the same column are replaced by the letter beneath, with the top element of the column following the last.

□ Otherwise, each plaintext letter is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter.

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

Figure 1.15Playfair

Plaintext = meet me at the school house

Splitting two letters as a unit => me et me at thesch ox ol ho us ex

Corresponding cipher text => CL KL CL RS PD IL HY AV MP HF XL IU

Strength of playfair cipher

□ Playfair cipher is a great advance over simple mono alphabetic ciphers.

□ Since there are 26 letters, 26x26 = 676 diagrams are possible, so identification of individual digram is more difficult.

□ Frequency analysis is much more difficult.

Hill cipher

This is proposed by Lester Hill in 1929. In this method, m letters together are substituted by m cipher letters. The formula is c = k * p (mod 26), where k is m × m matrix (entries are mod 26). p, c are column vectors of size m. To get plain text back we use the formula $k^{-1}c \pmod{26} = p$.

We now discuss the computation of A^{-1} in mod 26.

$$\text{Let } A = \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix}, \qquad |A| = 15 - 136 = -121 = 9 \pmod{26}$$

$$A^{-1} = \frac{1}{9}\begin{pmatrix} 3 & -8 \\ -17 & 5 \end{pmatrix} \bmod 26$$

1/9=3 (mod 26), since 3 and 9 are multiplicative inverses of mod 26. Thus,

$$A^{-1} = 3\begin{pmatrix} 3 & 18 \\ 9 & 5 \end{pmatrix}$$

$$= \begin{pmatrix} 9 & 54 \\ 27 & 15 \end{pmatrix} \bmod 26$$

$$= \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

Example: Let m=3.

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

Suppose that plain text is "pay more money". Take first three characters and find its cipher.

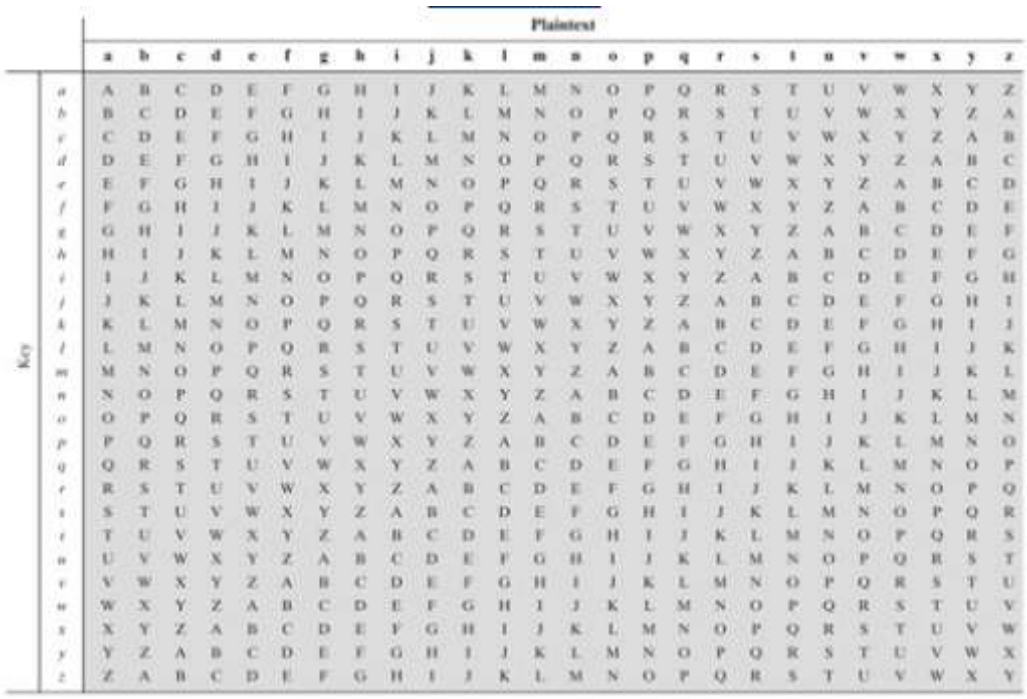
$$pay = \begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix}$$

$$c = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix} = \begin{pmatrix} 375 \\ 819 \\ 486 \end{pmatrix} \bmod 26 = \begin{pmatrix} 11 \\ 13 \\ 18 \end{pmatrix} = LNS.$$

Decryption requires k^{-1}.

$|k| = -939 = 23 \pmod{26}$

Vigenere cipher



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.

The strength of this cipher is that there are multiple ciphertext letters for each plaintext letter, one for each unique letter of the keyword. Thus, the letter frequency information is obscured.

Vigenere Cipher

This scheme of cipher uses a text string (say, a word) as a key, which is then used for doing a

number of shifts on the plaintext.

For example, let's assume the key is 'point'. Each alphabet of the key is converted to its respective numeric value: In this case,

p → 16, o → 15, i → 9, n → 14, and t → 20.

Thus, the key is: 16 15 9 14 20.

Process of Vigenere Cipher

- The sender and the receiver decide on a key. Say 'point' is the key. Numeric representation of this key is '16 15 9 14 20'.
- The sender wants to encrypt the message, say 'attack from south east'. He will arrange plaintext and numeric key as follows −

a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14

- He now shifts each plaintext alphabet by the number written below it to create ciphertext as shown below −

a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14
Q	I	C	O	W	A	U	A	C	G	I	D	D	H	B	U	P	B	H

- Here, each plaintext character has been shifted by a different amount – and that amount is determined by the key. The key must be less than or equal to the size of the message.
- For decryption, the receiver uses the same key and shifts received ciphertext in reverse order to obtain the plaintext.

Q	I	C	O	W	A	U	A	C	G	I	D	D	H	B	U	P	B	H
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14
a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t

Security Value

Vigenere Cipher was designed by tweaking the standard Caesar cipher to reduce the effectiveness of cryptanalysis on the ciphertext and make a cryptosystem more robust. It is significantly **more secure than a regular Caesar Cipher**.

One Time Pad Cipher

It is an unbreakable cryptosystem.

The circumstances are −

- The length of the keyword is same as the length of the plaintext.
- The keyword is a randomly generated string of alphabets.
- The keyword is used only once.

It represents the message as a sequence of 0s and 1s. this can be accomplished by writing all numbers in binary, for example, or by using ASCII. The key is a random sequence of 0□s and 1□s of same length as the message. Once a key is used, it is discarded and never used again. The system can be expressed as follows:

Ci = Pi ⊕ Ki

Ci - ith binary digit of cipher text

Pi - ith binary digit of plaintext

Ki - ith binary digit of key

⊕ – exclusive OR opearaiton

Thus the cipher text is generated by performing the bitwise XOR of the plaintext and the key. Decryption uses the same key. Because of the properties of XOR, decryption simply involves the same bitwise operation:

Pi = Ci⊕ Ki

e.g., plaintext = 0 0 1 0 1 0 0 1

Key = 1 0 1 0 1 1 0 0

---

ciphertext = 1 0 0 0 0 1 0 1

Advantage:

 Encryption method is completely unbreakable for a ciphertext only attack.

Disadvantages

 It requires a very long key which is expensive to produce and expensive to transmit.

 Once a key is used, it is dangerous to reuse it for a second message; any knowledge on the first message would give knowledge of the second.

TRANSPOSITION TECHNIQUES

All the techniques examined so far involve the substitution of a cipher text 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.

(i) Rail fence

It is simplest of such cipher, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

Plaintext = "meet me after the toga party"

To encipher this message with a rail fence of depth 2, we write the message as follows:

```
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

(ii) simple columnar transposition

An example is a 'simple columnar transposition' cipher where the plaintext is written horizontally with a certain alphabet width. Then the ciphertext is read vertically as shown.

For example, the plaintext is "golden statue is in eleventh cave"

and the secret random key chosen is "five". We arrange this text horizontally in table with number of column equal to key value. The resulting text is shown below.

g	o	l	d	e
n	s	t	a	t
u	e	i	s	i
n	e	l	e	v
e	n	t	h	c
a	v	e		

The ciphertext is obtained by reading column vertically downward from first to last column.

The ciphertext is 'gnuneaoseenvltiltedasehetivc'.

To decrypt, the receiver prepares similar table. The number of columns is equal to key number. The number of rows is obtained by dividing number of total ciphertext alphabets by key value and rounding of the quotient to next integer value.

The receiver then writes the received ciphertext vertically down and from left to right column. To obtain the text, he reads horizontally left to right and from top to bottom row.

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is more complex permutation that is not easily reconstructed.

STEGANOGRAPHY

A plaintext message may be hidden in any one of the 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, is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message.

e.g., (i) the sequence of first letters of each word of the overall message spells out the real (hidden) message. (ii) Subset of the words of the overall message is used to convey the hidden message.

Various other techniques have been used historically, some of them are

- Character marking – selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held to 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 in front of the light.

- Typewritten correction ribbon – used between the lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Drawbacks of steganography

- Requires a lot of overhead to hide a relatively few bits of information.

 Once the system is discovered, it becomes virtually worthless.

Rotor Machines

The example just given suggests that multiple stages of encryption can produce an algorithm that is significantly more difficult to cryptanalyze. This is as true of substitution ciphers as it is of transposition ciphers. Before the introduction of DES, the most important application of the principle of multiple stages of encryption was a class of systems known as rotor machines.

The basic principle of the rotor machine is illustrated in Figure . 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.

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 , 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 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 keystroke. The right half of Figure 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.

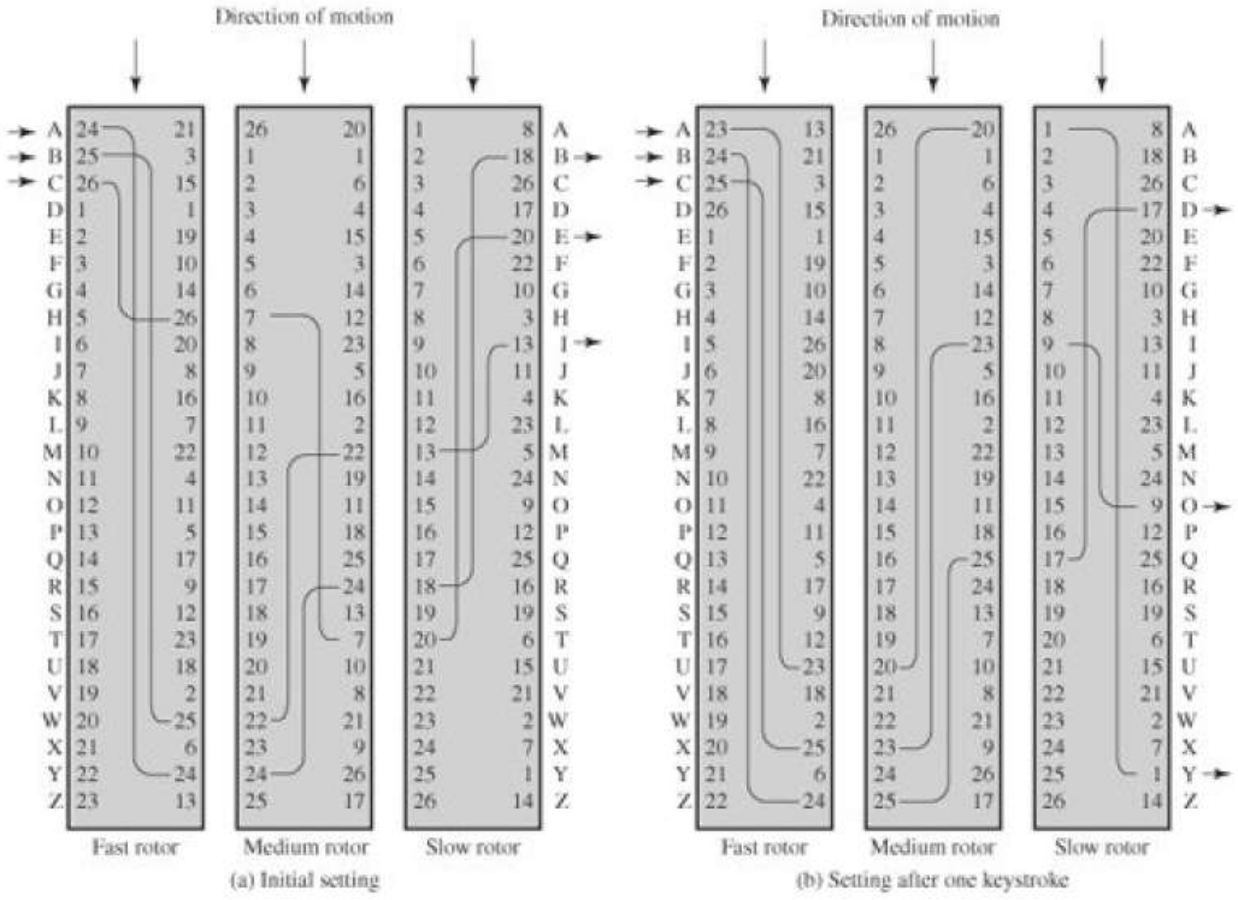


Figure 1.12 Three-Rotor Machine with Wiring Represented by Numbered Contacts

POSSIBLE QUESTIONS

PART A (20 x 1 = 20 marks)

(ONLINE EXAMINATION)

PART B (5 x 6 = 30 marks)

1. Explain the following
   i) Differentiate passive and active attacks
   ii) Differentiate a block cipher and a stream cipher?
2. Explain in detail about Security Services and OSI architecture
3. Explain the following
   i) Transposition Techniques
   ii) A model for network security
4. List and explain any two Substitution Techniques with example.
5. Briefly describe Security Attacks, Symmetric and Asymmetric Cryptography
6. i) List and Define Substitution Techniques?
   ii) Given the key "MONARCHY" apply play fair to plain text "FACTIONALISM" to ensure confidentially at the destination, decrypt the cipher text and establish authenticity.
7. Write note on Rotor machines and Transposition Techniques.
8. Illustrate in detail about the categories of security services for X.800?
9. i) Write a brief note on a model for network security.
   ii) Explain about Steganography.
10. i) Explain in detail about the OSI security architecture?
    ii) Give note on Symmetric Cipher Model

PART – C (1 x 10 = 10 marks)

1. User Bob sends the following message to user Alice. Calculate the encrypted message using playfair cipher by creating a key table:
The message is : “hide money”
The sender Bob and the receiver Alice decide on a particular key, ‘tutorials’.

2. To pass an encrypted message from one person to another, it is first necessary that both parties have the 'key' for the cipher, so that the sender may encrypt it and the receiver may decrypt it. Perform the encryption steps involved with the Caesar cipher.
Encrypt the following text:
   a. 'defend the east wall of the castle',

Shift (key) = 3.

b. 'the other side of silence'

Shift (key) = 5.

3. Explain Substitution Techniques

4. Describe about Transposition Techniques.

5. Explain OSI security architecture.

KARPAGAM ACADEMY OF HIGHER EDUCATION

DEPARTMENT OF COMPUTER SCIENCE

I M.Sc CS

CRYPTOGRAPHY AND NETWORK SECURITY

	UNIT 1					
S.NO	**Question**	**Option 1**	**Option 2**	**Option 3**	**Option 4**	**Answer**
1	CMAC stands for	cipher based mask authentication code	cipher based message	common based message	cipher based message	cipher based message
2	_____ is the study of techniques for ensuring the secrecy and/or authenticity of information	cryptology	network security	computer security	internet	cryptology
3	_____ which deals with the defeating such techniques to recover information	computer security	network security	cryptanalysis	internet	cryptanalysis
4	_____ area covers the use of cryptographic algorithms in network protocol and network application	computer security	network security	internet	mobile security	network security
5	_____ term to refer to the security of computers against intruders.	internet	network security	computer security	mobile security	computer security
6	the generic name for the collection of tools designed to protect data and to thwart hackers is_____	computer security	network security	internet	mobile security	computer security
7	_____ measures are needed to protect data during their transmission	mobile security	computer security	internet	network security	network security
8	IAB stands for _____	Intel Arithmetic Board	Internet Arithmetic Board	Internet Architecture Board	Intel Architecture	Internet Architecture Board
9	CERT stands for_____	computer electric response team	computer emergency	computer electric reply team	computer electric reply	computer emergency
10	the _____ security architecture is useful to managers as a way of organizing the task of providing security	OSI	SIO	OSII	OSA	OSI
11	the OSI architecture focuses on security _____, _____ and _____	internet,base,security	net,mech,none	attack,internet,services	attack,mechanism,service	attack,mechanism,service
12	Any action that compromises the security of information owned by an organization is known	Security attack	Security Service	Security mechanism	None	Security attack

13	. _____is a mechanism that is designed to detect,prevent or recover from a security attack	Security service	Security attack	Security mechanism	None	Security mechanism
14	_____ is a service that enhances the security of the data processing systems and the information transfers	Security mechanism	Security service	Security attack	None	Security service
15	_____ are in the nature of eavesdropping on,or monitoring of, transmissions	mobile attack	active attacks	internet attacks	passive attacks	passive attacks
16	_____ are very difficult to detect because they do not involve any alteration of the data	active attacks	passive attacks	internet attacks	mobile attack	passive attacks
17	_____involves some modification of the data stream or the creation of a false stream.	active attacks	passive attacks	internet attacks	mobile attack	active attacks
18	A _____ takes place when one entity pretends to be a different entity.	Reply	masquerade	denial of service	X.800	masquerade
19	_____ involves the passive capture of a data unit and its subsequent retransmission to produce an	Reply	masquerade	denial of service	X.801	Reply
20	the ____ prevents or inhibits the normal use or management of communication facilities.	masquerade	denial of service	Reply	X.802	denial of service
21	_____ defines a security service as a service provided by a protocol layer of communicating open	X.600	X.500	X.800	X.8000	X.800
22	X.800 divides services into_____categories and _____specific services	5,14	6,14	5,19	6,19	5,14
23	._____requires the access to information reduces may be controlled by or for the target system	authentication	Non repudiation	Access control	confidentiality	Access control
24	_____ requires that neither the sender nor the receiver of the message be able to deny the	Non repudiation	Authentication	Access control	Access control	Non repudiation
25	_____requires that computer system assets be available to authorized parties when needed.	Non repudiation	Authentication	Availability	Access control	Availability
26	_____ is the protection of transmitted data from passive attacks	Authentication	confidentiality	Non repudiation	Access control	confidentiality
27	_____ provides for the corroboration of the source of a data unit	data origin authentication	peer entity authentication	transformation	Access control	data origin authentication
28	_____ exploit service flaws in computer to inhibit use by legitimate users	service threats	information threats	internet	network security	service threats
29	_____ threats intercepts or modify data on behalf of users who should not have access to that data	internet	service threats	information access	network security	information access
30	_____ and _____ are two examples of software attacks	viruses and worms	service	internet	network	viruses and worms
31	_____ encryption is a form of cryptosystem in which encryption and decryption are performed using the	symmetric	asymmetric	service	network	symmetric
32	_____encryption transforms plaintext into ciphertext using a secret key and an encryption algorithms	service	asymmetric	symmetric	network	symmetric

33	_____ involves trying all possible keys	brute force	symmetric	asymmetric	network	brute force
34	_____ techniques map plaintext elements into ciphertext elements	transposition	substitution	service	network	substitution
35	_____ techniques systematically transpose the positions of plaintext elements	transposition	substitution	service	network	transposition
36	_____machines are sophisticated precomputer hardware devices that use substitution techniques	computer	rotor	network	embedded	rotor
37	_____ is a technique for hiding a secret message within a larger one	steganography	encryption	decryption	none	steganography
38	An original message is known as_____	ciphertext	input	plaintext	output	plaintext
39	the coded message is called the_____	plaintext	ciphertext	input	output	ciphertext
40	the process of converting from plain text into ciphertext is known as_____	enciphering	deciphering	substitution	transposition	enciphering
41	restoring the plaintext from the ciphertext is_____	deciphering	enciphering	substitution	transposition	deciphering
42	the _____ key is also input to the encryption algorithm	secret	plain	cipher	none	secret
43	_____is essentially the encryption algorithm run in reverse	substitution	encryption	decryption	transposition	decryption
44	the ____ algorithm performs various substition and transformation on the plaintext	transposition	encryption	substitution	decryption	encryption
45	_____is the scrambled message produced as output	plaintext	ciphertext	input	output	ciphertext
46	_____ types of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext	transposition	substitution	cryptanalysis	none	cryptanalysis
47	A ____ techniques is one in which the letter of plaintext are replaced by other letters or by numbers or symbols	substitution	transposition	cryptanalysis	none	substitution
48	the _____cipher involves replacing each letter of the alphabet with the letter standing three places further	caesar	monoalphabetic cipher	playfair cipher	hill ciphers	caesar
49	_____,the relative frequency of the letters can be determined and compared to a standard frequency	playfair cipher	hill ciphers	monoalphabetic cipher	caesar	monoalphabetic cipher
50	____ treats diagrams in the plaintext as single units and translates these units into ciphertext diagrams	playfair cipher	monoalphabetic cipher	hill ciphers	caesar	playfair cipher
51	_____ takes m successive plaintext letters and substitutes for them m ciphertext letters	playfair cipher	monoalphabetic cipher	caesar	hill ciphers	hill ciphers
52	_____produces random output that bears no statistical relationship to the plaintext	one time pad	vigenere	rail fence	none	one time pad

53	_____ which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows	one time pad	rail fence	vigenere	none	rail fence
54	_____selected letters of printed or typewritten text are overwritten in pencil	pin punctures	invisible ink	character marking	type writer correction	character marking
55	_____a number of substances can be used for writing but leave no visible trace until heat	character marking	invisible ink	pin punctures	type writer correction	invisible ink
56	small _____ on selected letters are ordinarily not visible unless the paper is held up in front of a light	pin punctures	type writer correction ribbon	character marking	pin punctures	pin punctures
57	_____used between lines typed with a black ribbon	character marking	pin punctures	invisible ink	type writer correction	type writer correction ribbon
58	the machines consists of a set of independently rotating cylinders through which_____ can flow	electrical pulse	waves	rail fence	none	electrical pulse
59	_____,in which a keyword is concatenated with the plaintext itself to provide a running key	autokey system	random key system	rail fence	none	autokey system
60	the best known and one of the simplest such algorithm is referred to as the _____ cipher	vigenere	rail fence	one time pad	none	vigenere

UNIT –II

Syllabus

Block Ciphers and Data Encryption Standard –Block Cipher Principles – The Data Encryption Standard - The Strength of DES –Advanced Encryption Standard (AES) – Evaluation Criteria for AES – The AES Cipher – Multiple Encryption and Triple DES – Block Cipher Modes of Operation – Stream Ciphers and RC4- modular Arithmetic and Euclidean Algorithm.

BLOCK CIPHERS AND DATA ENCRYPTION STANDARD

CIPHER PRINCIPLES

Virtually, all symmetric block encryption algorithms in current use are based on a structure referred to as Fiestel block cipher. For that reason, it is important to examine the design principles of the Fiestel cipher. We begin with a comparison of stream cipher with block cipher**.**

• **A stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. E.g, vigenere cipher.

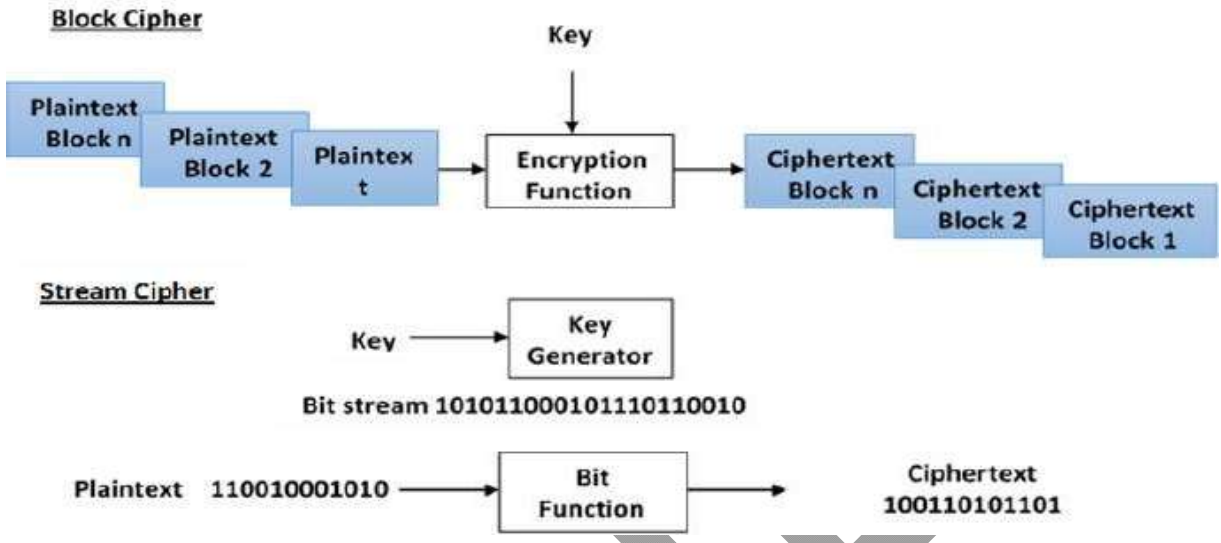
• **A block cipher** is one in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically a block size of 64 or 128 bits is used.

Block Ciphers

In this scheme, the plain binary text is processed in blocks (groups) of bits at a time; i.e. a block of plaintext bits is selected, a series of operations is performed on this block to generate a block of ciphertext bits. The number of bits in a block is fixed. For example, the schemes DES and AES have block sizes of 64 and 128, respectively.

Stream Ciphers

In this scheme, the plaintext is processed one bit at a time i.e. one bit of plaintext is taken, and a series of operations is performed on it to generate one bit of ciphertext. Technically, stream ciphers are block ciphers with a block size of one bit.



Block cipher principles

Most symmetric block ciphers are based on a Feistel Cipher Structure.It is needed since most be able to decryptciphertext to recover messages efficiently Block ciphers look like an extremely large substitution. It would need table of 264 entries for a 64-bit block. Itis created from smaller building blocks. It was the using idea of a product cipher.

In 1949 Claude Shannon introduced idea of substitution-permutation (S-P) networks called modern substitution-transposition product cipher these form the basis of modern block ciphers

• S-P networks are based on the two primitive cryptographic operations we have seen before:

• *substitution*(S-box)

• *permutation*(P-box)

• **Substitution:** Each plaintext element or group of elements is uniquely replaced by a corresponding cipher text element 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.

• provide *confusion* and *diffusion* of message

• **diffusion**– dissipates statistical structure of plaintext over bulk of ciphertext

• **confusion**– makes relationship between ciphertext and key as complex as possible

FEISTEL CIPHER STRUCTURE

Feistel Cipher is not a specific scheme of block cipher. It is a design model from which many different block ciphers are derived. DES is just one example of a Feistel Cipher. A cryptographic system based on Feistel cipher structure uses the same algorithm for both encryption and decryption.

Encryption Process

The encryption process uses the Feistel structure consisting multiple rounds of processing of the plaintext, each round consisting of a "substitution" step followed by a permutation step.

Feistel Structure is shown in the following illustration –

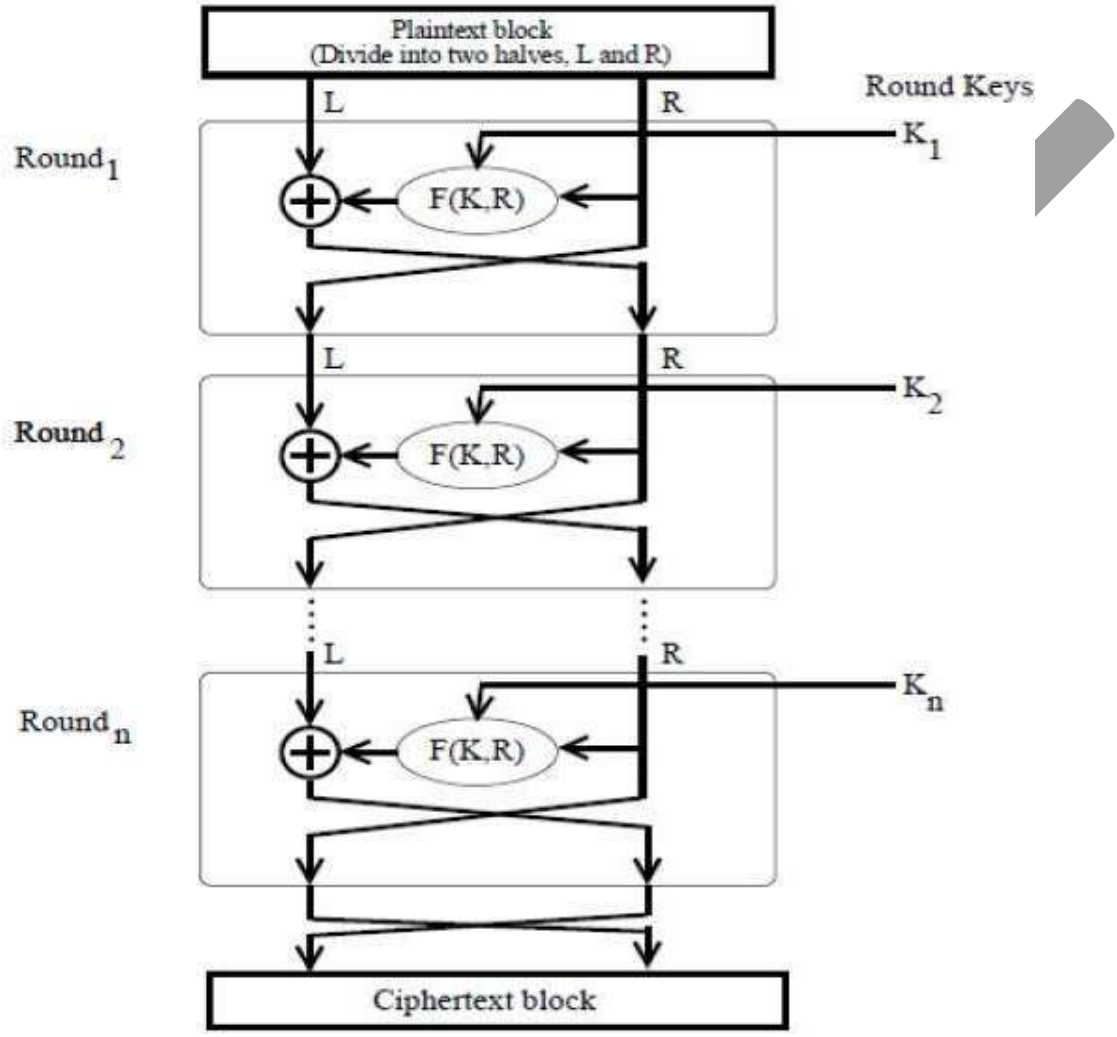


Figure 2.1 Classical Feistel Network

- The input to the encryption algorithm are a plaintext block of length 2w bits and a key K. The input block to each round is divided into two halves that can be denoted as L and R for the left half and the right half.

- In each round, the right half of the block, R, goes through unchanged. But the left half, L, goes through an operation that depends on R and the encryption key. First, we apply an encrypting function 'f' that takes two input − the key K and R. The function produces the output f(R,K). Then, we XOR the output of the mathematical function with L.

- In real implementation of the Feistel Cipher, such as DES, instead of using the whole encryption key during each round, a round-dependent key (a subkey) is derived from the encryption key. This means that each round uses a different key, although all these subkeys are related to the original key.

- The permutation step at the end of each round swaps the modified L and unmodified R. Therefore, the L for the next round would be R of the current round. And R for the next round be the output L of the current round.

- Above substitution and permutation steps form a 'round'. The number of rounds are specified by the algorithm design.

- Once the last round is completed then the two sub blocks, 'R' and 'L' are concatenated in this order to form the ciphertext block.

The difficult part of designing a Feistel Cipher is selection of round function 'f'. In order to be unbreakable scheme, this function needs to have several important properties that are beyond the scope of our discussion.

Decryption Process

The process of decryption in Feistel cipher is almost similar. Instead of starting with a block of plaintext, the ciphertext block is fed into the start of the Feistel structure and then the process thereafter is exactly the same as described in the given illustration.

The process is said to be almost similar and not exactly same. In the case of decryption, the only difference is that the subkeys used in encryption are used in the reverse order.

The final swapping of 'L' and 'R' in last step of the Feistel Cipher is essential. If these are not swapped then the resulting ciphertext could not be decrypted using the same algorithm.

Number of Rounds

The number of rounds used in a Feistel Cipher depends on desired security from the system. More number of rounds provide more secure system. But at the same time, more rounds mean the

inefficient slow encryption and decryption processes. Number of rounds in the systems thus depend upon efficiency–security tradeoff.

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

- **Block size** - Increasing size improves security, but slows cipher
- **Key size** - Increasing size improves security, makes exhaustive key searching harder, but may slow cipher
- **Number of rounds** - Increasing number improves security, but slows cipher
- **Subkey generation** - Greater complexity can make analysis harder, but slows cipher
- **Round function** - Greater complexity can make analysis harder, but slows cipher
- **Fast software en/decryption & ease of analysis -** are more recent concerns for practical use and testing.

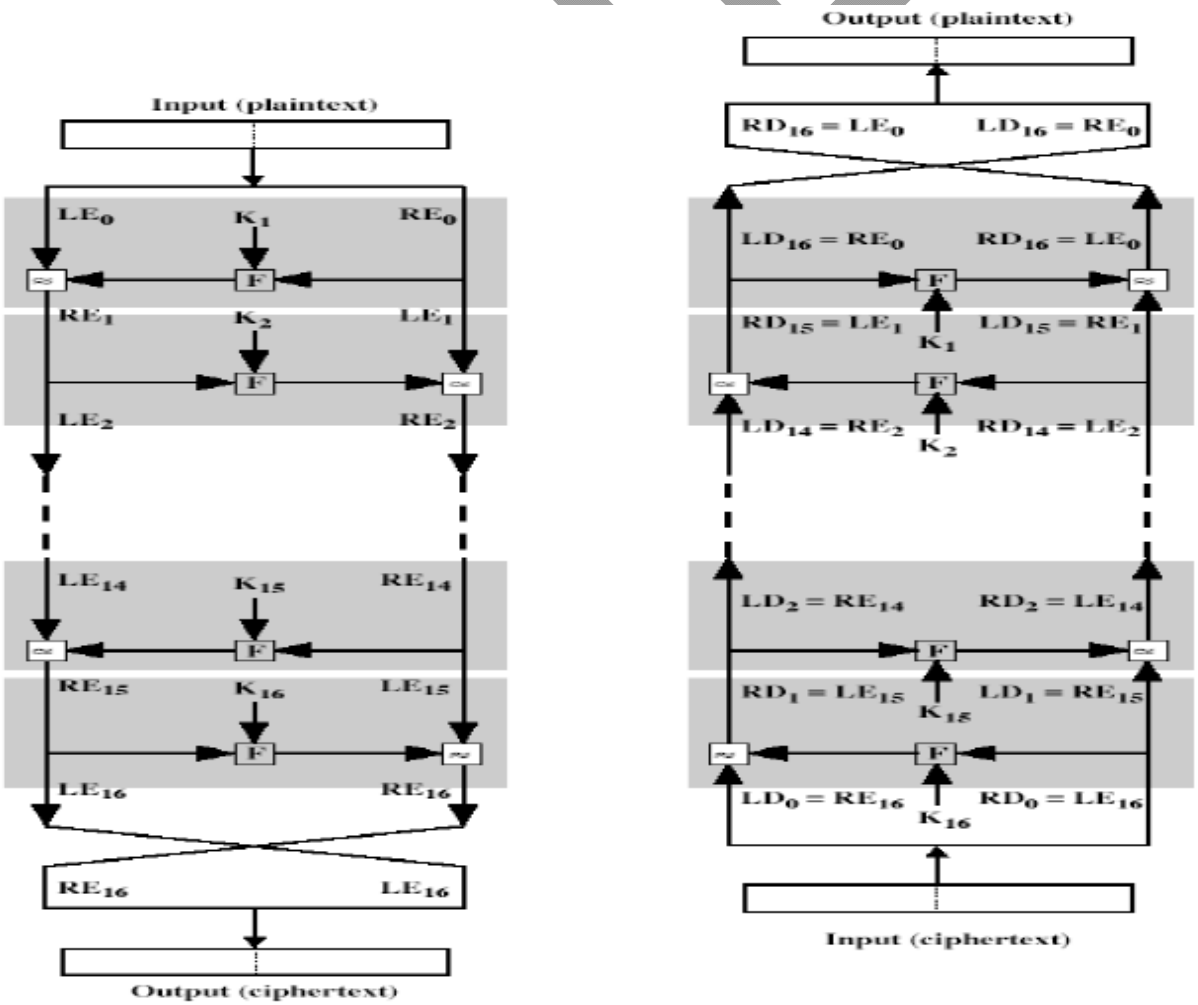


Figure 2.2 Feistel Encryption and Decryption

The process of decryption is essentially the same as the encryption process. The rule is as follows: use the cipher text as input to the algorithm, but use the subkeyki in reverse order. i.e., kn in the first round, kn-1 in second round and so on. For clarity, we use the notation LEi and REi for data traveling through the decryption algorithm. The diagram below indicates that, at each round, the intermediate value of the decryption process is same (equal) to the corresponding value of the encryption process with two halves of the value swapped. i.e., REi || LEi (or) equivalently RD16-i || LD16-i

After the last iteration of the encryption process, the two halves of the output are swapped, so that the cipher text is RE16 || LE16. The output of that round is the cipher text. Now take the cipher text 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.

Now we will see how the output of the first round of the decryption process is equal to a 32-bit swap of the input to the sixteenth round of the encryption process.

First consider the encryption process,

LE16 = RE15

RE16 = LE15 F (RE15, K16)

On the decryption side, LD1 =RD0 = LE16 =RE15

RD1 = LD0 F (RD0, K16)

= RE16 F (RE15, K16)

= [LE15 F (RE15, K16)] F (RE15, K16)

= LE15

Therefore,

LD1= RE15

RD1 = LE15

In general, for the ith iteration of the encryption algorithm,

LEi = REi-1 REi = LEi-1 F (REi-1, Ki)

Finally, the output of the last round of the decryption process is RE0 || LE0. A 32-bit swap recovers the original plaintext.

DATA ENCRYPTION STANDARD (DES)

DES was the most widely used block cipher in world adopted in 1977 by NBS (now NIST). It encrypts 64-bit data using 56-bit key. It has widespread use has been considerable controversy over its security.

DES History

IBM developed Lucifer cipher by team led by Feistel. It used 64-bit data blocks with 128-bit key. It was then redeveloped as a commercial cipher with input from NSA and others. In 1973 NBS issued request for proposals for a national cipher standard. IBM submitted their revised Lucifer which was eventually accepted as the DES.

DES Design Controversy

Although DES standard is public, it was considerable controversy over design and in choice of 56-bit key (vs Lucifer 128-bit). DES has become widely used, especially in financial applications

DES Encryption

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST).

DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. Though, key length is 64-bit, DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only). General Structure of DES is depicted in the following illustration −

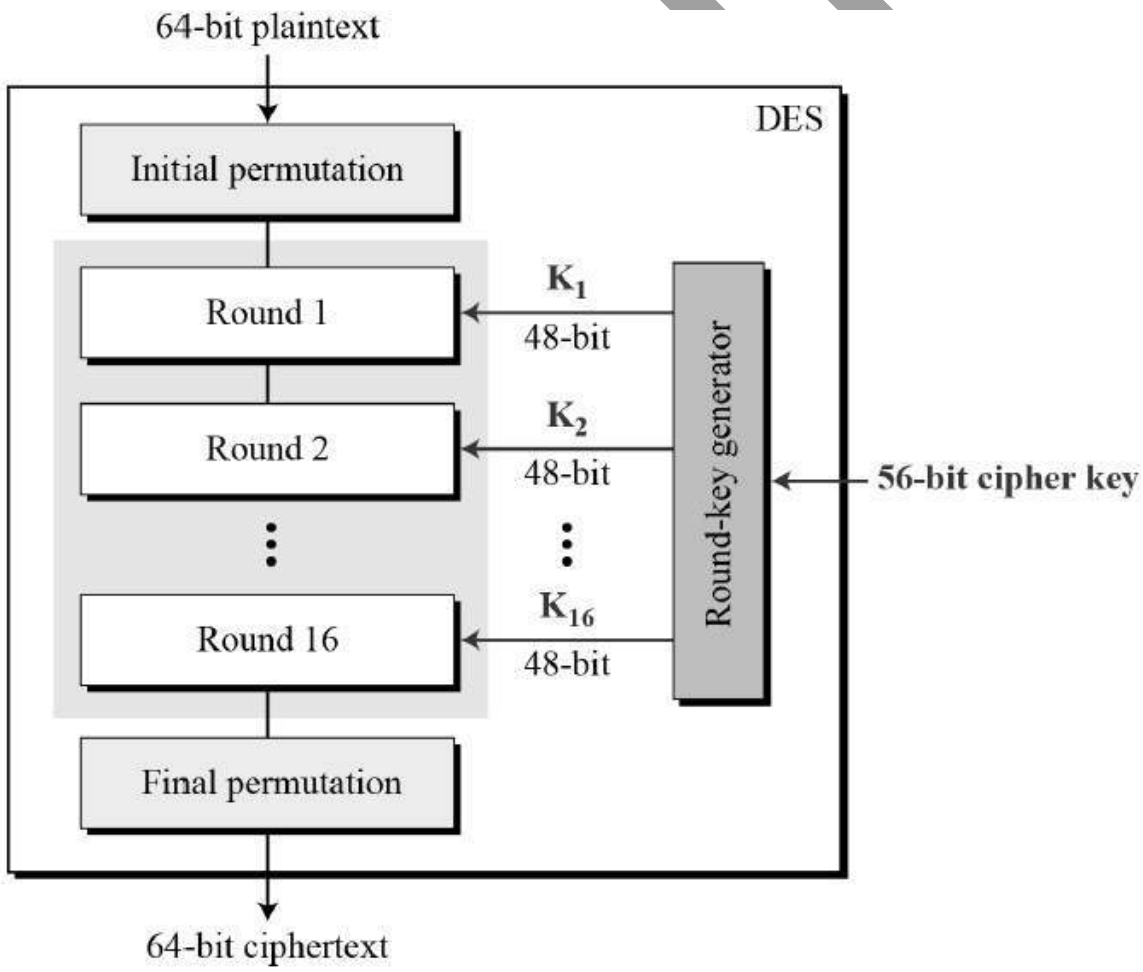


Fig 2.3 DES Model

Since DES is based on the Feistel Cipher, all that is required to specify DES is −

- Initial & Final Permutation

- Round function
- Key schedule
- Any additional processing − Initial and final permutation

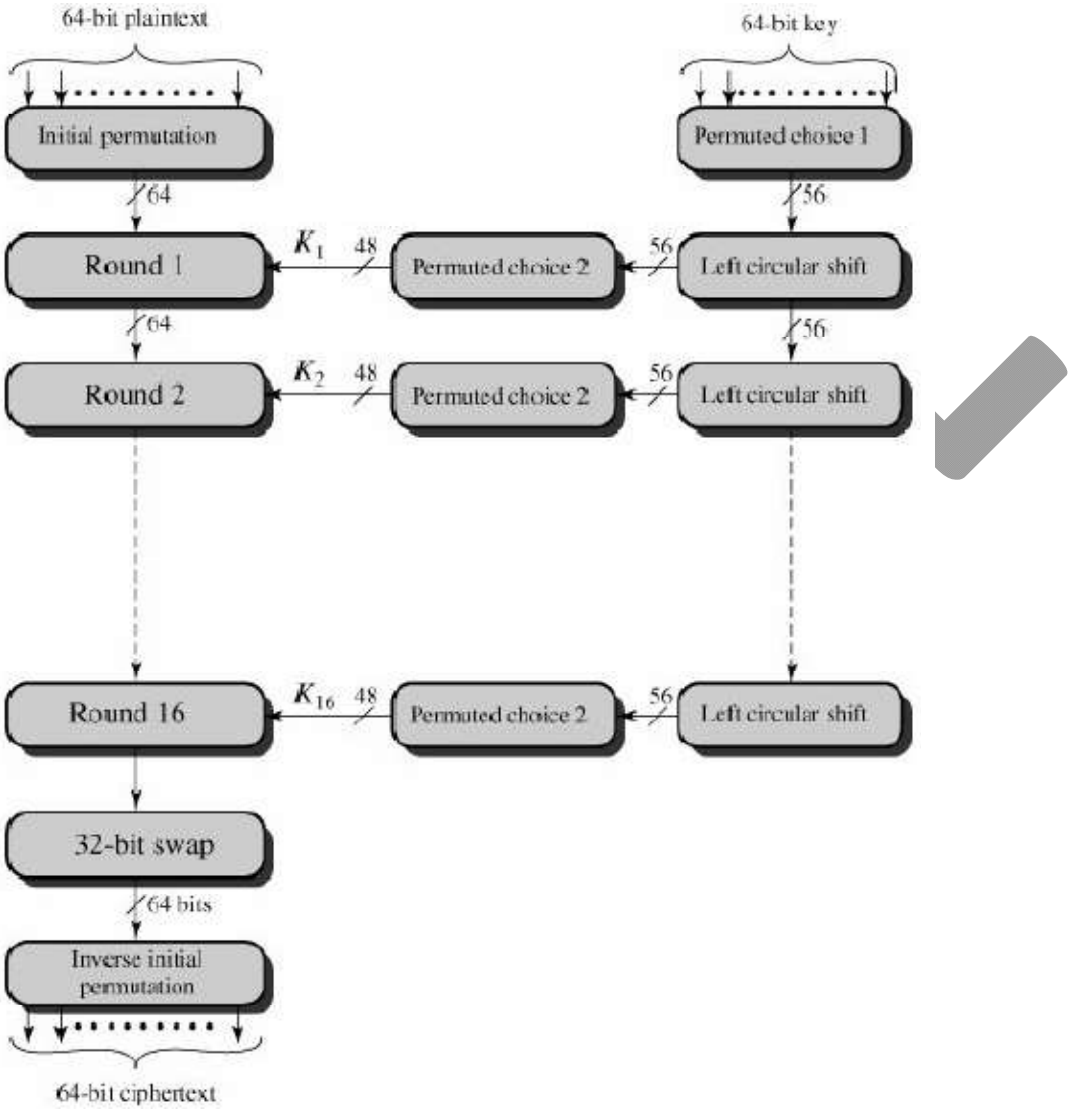


Fig 2.3 DES RoundKey Model

Initial and Final Permutation

The initial and final permutations are straight Permutation boxes (P-boxes) that are inverses of each other. They have no cryptography significance in DES. The initial and final permutations are

shown as follows −

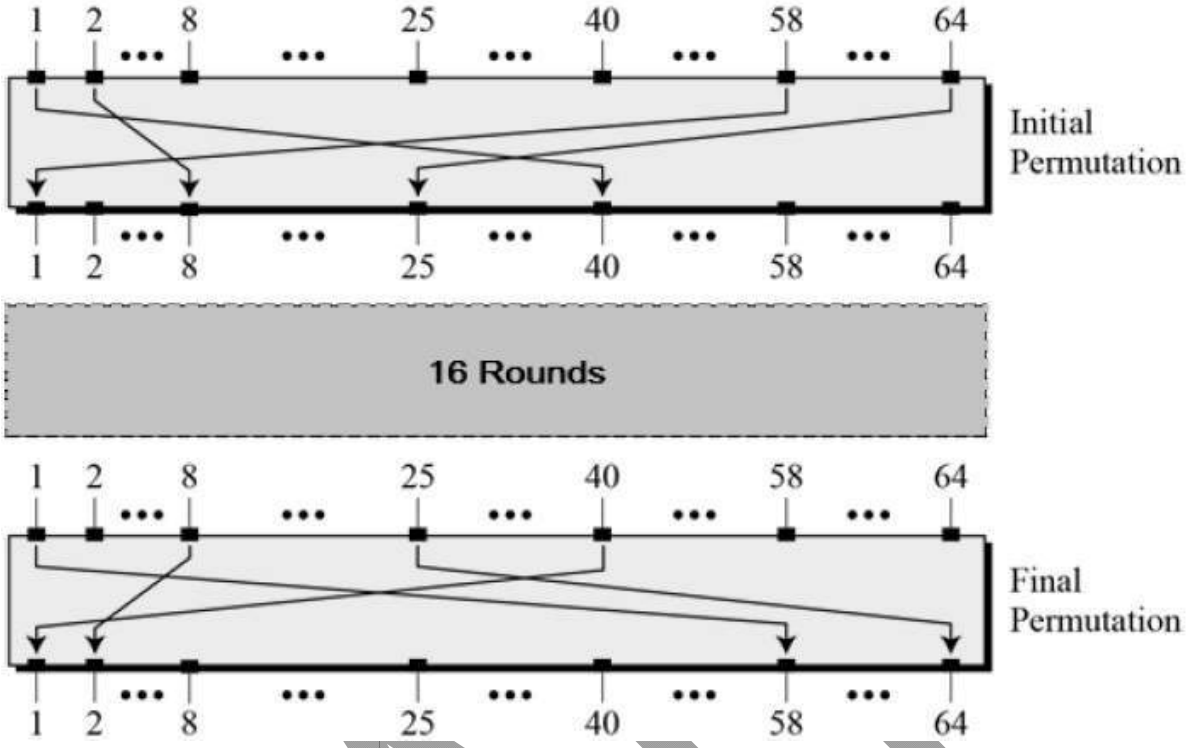


Fig 2.4. Initial and Final Permutation

Round Function

The heart of this cipher is the DES function, *f*. The DES function applies a 48-bit key to the rightmost 32 bits to produce a 32-bit output.

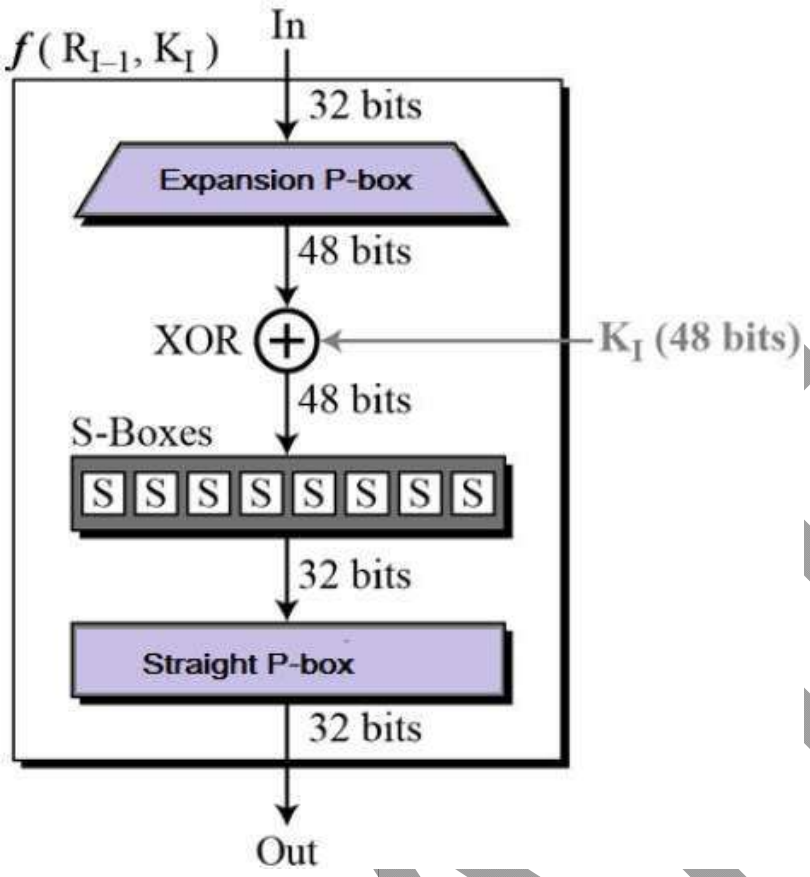


Fig 2.5. Single Round Function

- **Expansion Permutation Box** − Since right input is 32-bit and round key is a 48-bit, we first need to expand right input to 48 bits. Permutation logic is graphically depicted in the following illustration −

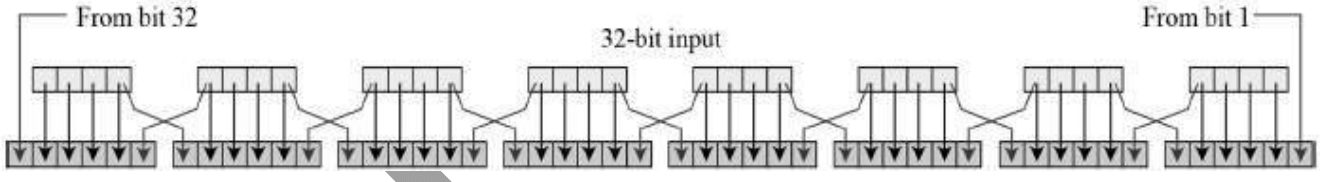


Fig 2.6 Expansion Box

- The graphically depicted permutation logic is generally described as table in DES specification illustrated as shown −

32	01	02	03	04	05
04	05	06	07	08	09
08	09	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	31	31	32	01

- **XOR (Whitener).** − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.
- **Substitution Boxes.** − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration −

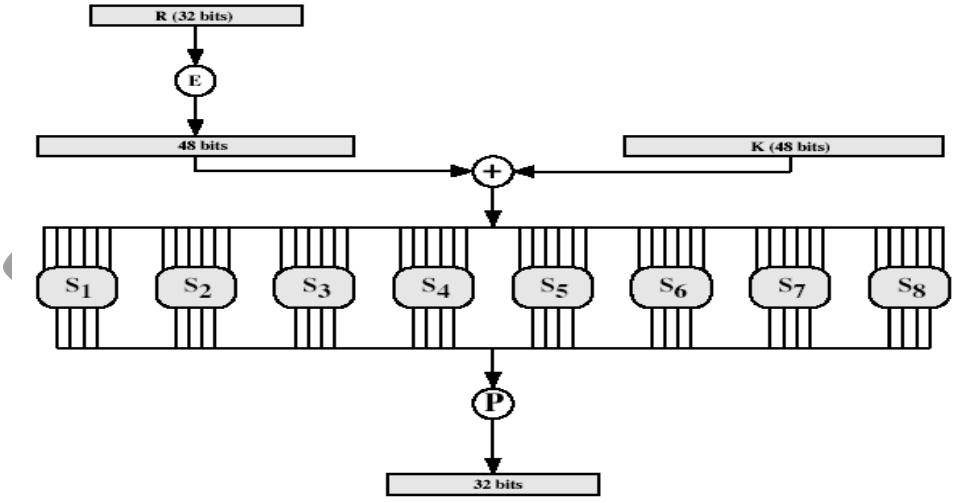


Fig 2.7 Substitution Box

Substitution Boxes S

They have eight S-boxes which map 6 to 4 bits. Each S-box is actually 4 bit boxes. The outer bits 1 & 6 (row bits) select one rows inner bits 2-5 (col bits) are substituted. The result is 8 lots of 4 bits, or 32 bits. The row selection depends on both data & key. This feature is known as autoclaving (autokeying)

- **Straight Permutation** − The 32 bit output of S-boxes is then subjected to the straight permutation with rule shown in the following illustration:

16	07	20	21	29	12	28	17
01	15	23	26	05	18	31	10
02	08	24	14	32	27	03	09
19	13	30	06	22	11	04	25

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key.

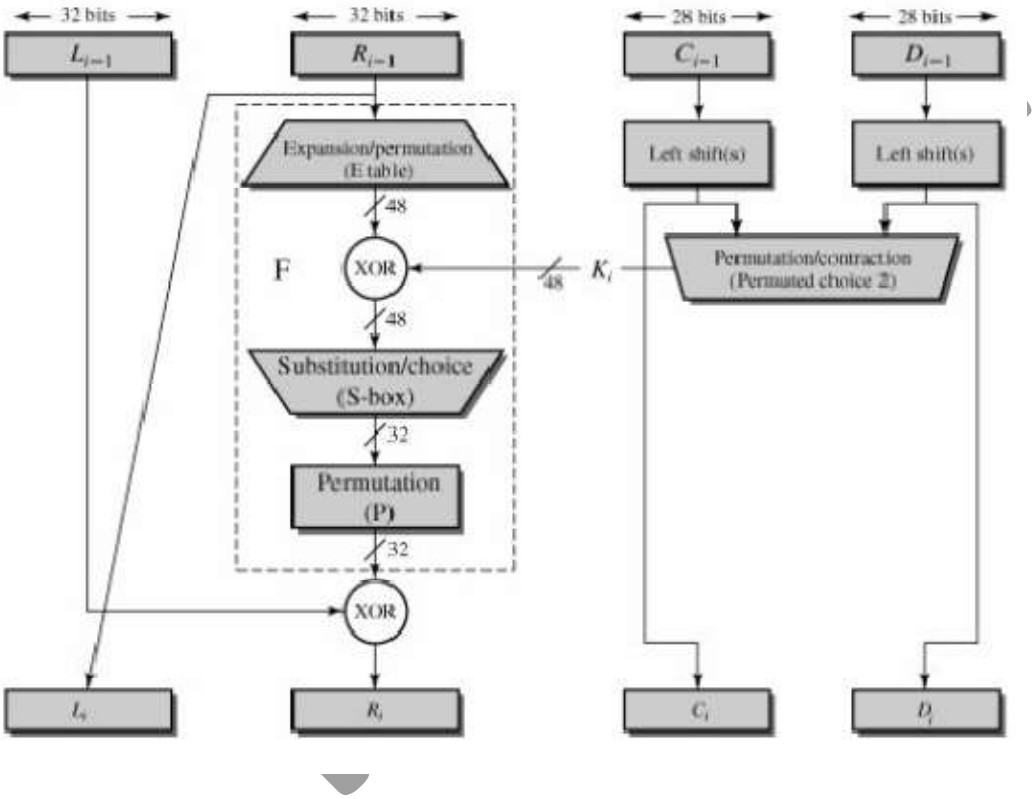


Fig 2.8. DES Key generation

DES Key Schedule

This forms subkeys used in each round. It consists of: initial permutation of the key (PC1) which selects 56-bits in two 28-bit halves. The 16 stages consisting of:

a) selecting 24-bits from each half

b) permuting them by PC2 for use in function f,

c) rotating each half separately either 1 or 2 places depending on the key rotation schedule K

DES Decryption

DES decryption must unwind steps of data computation with Feistel design, do encryption steps again.

Using subkeys reverse the order (SK16 … SK1). Note that IP undoes final FP step of encryption. First round with SubKey16 undoes 16th encryption round. Sixteenth round with SK1 undoes 1st encryption round. Then final FP undoes initial encryption IP. Thus recovering the original data value.

Avalanche Effect

This is the key desirable property of encryption algorithm, where a change of one input or key bit results in changing approximately half output bits. Thus it makes attempts to "home-in" by guessing keys impossible. DES exhibits strong avalanche.

Strength of DES

Strength of DES – Key Size

It has56-bit keys which will have $2^{56} = 7.2 \times 10^{16}$ values. Brute force search looks hard, recent advances have shown is possible. In 1997 it was used on Internet in a few months. In 1998 it was used on dedicated h/w (EFF).

Strength of DES – Timing Attacks

Timing attack is the use of knowledge of consequences of implementation to derive knowledge of some/all subkey bits. This specifically use the fact that calculations can take varying times depending on the value of the inputs to it, particularly problematic on smartcards

Strength of DES – Analytic Attacks

Now there are several analytic attacks on DES. These utilise some deep structure of the cipher, by gathering information about encryptions or can eventually recover some/all of the sub-key bits, if necessary then exhaustively search for the rest. Generally these are statistical attacks, which includes

- differential cryptanalysis
- linear cryptanalysis
- related key attacks

Advanced Encryption Standard (AES)

AES Cipher-Rijendael

Designed by Rijmen-Daemen in Belgium
has 128/192/256 bit keys, 128 bit data an
iterative rather than feistel cipher
O treats data in 4groups of 4 bytes

- o operates an entire block in every round
- designed to be:
  - o resistant against known attacks
  - o speed and code compactness on many CPUs
  - o designs implicity
- processes data as 4groups of 4
- bytes(state)has 9/11/13 rounds in which state undergoes:
  - o byte substitution (1 S-box used on every byte)
  - o shift rows(permute bytesbetweengroups/columns)omixcolumns(subsusing matrixmultipyofgroups)oadd round key(XOR state with
- key material)
- initial XOR key material & incomplete last round

all operations can be combined into XOR and table lookups- hence very fast & efficient

The features of AES are as follows −

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java
- Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix −
- Unlike DES, the number of rounds in AES is variable and depends on the length of the key. AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key.

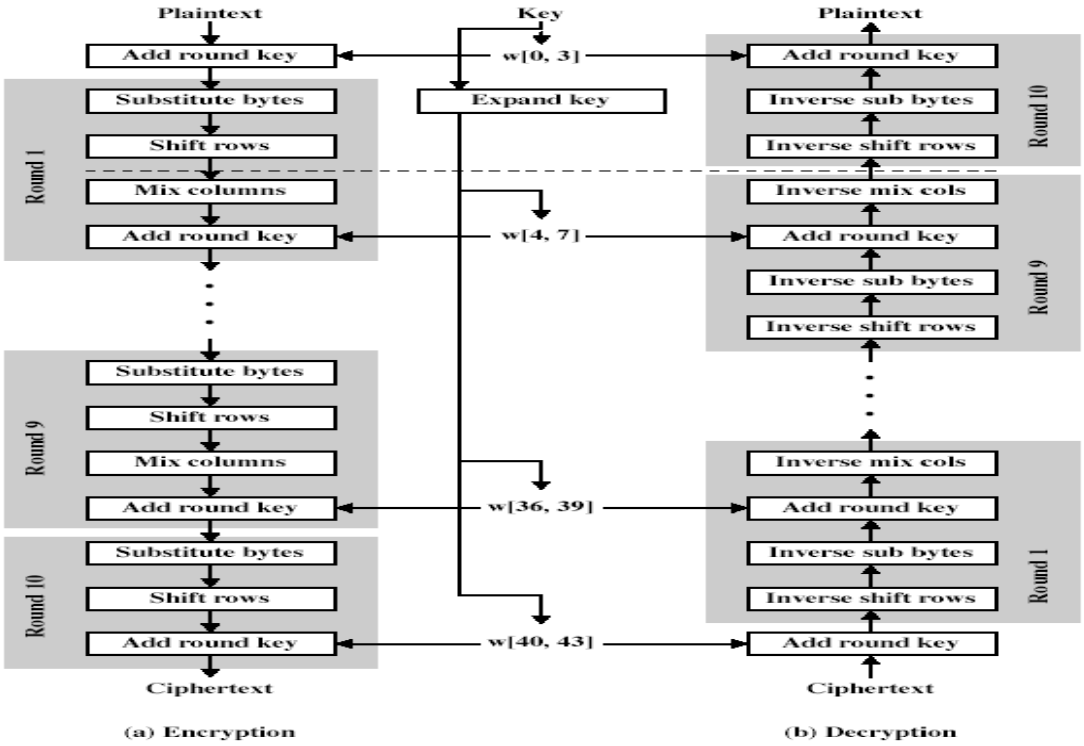


Fig 2.9 AES Round

- **Byte Substitution**

The 16 input bytes are substituted by looking up a fixed table (S-box) given in design. The result is in a matrix of four rows and four columns.

O a simple substitution of each byte
O uses one table of 16x16 bytes containing a permutation of all 256 8-bit values
O each byte of state is replaced by byte in row(left 4-bits)&column (right 4-bits)
O eg. byte {95} is replaced by row 9 col 5 byte o
which is the value {2A}
O S-box is constructed using a defined transformation of the values in $GF(2^8)$
O designed to be resistant to all known attacks

Shift Rows

Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are re-inserted on the right side of row. Shift is carried out as follows −

- First row is not shifted.
- Second row is shifted one (byte) position to the left.
- Third row is shifted two positions to the left.
- Fourth row is shifted three positions to the left.
- The result is a new matrix consisting of the same 16 bytes but shifted with respect to each other.

O a circular byte shift in each

. 1st row is unchanged

. 2nd row does 1 byte circular shift to left

□3rd row does 2 byte circular shift to left

□4th row does 3 byte circular shift to left

O decrypt does shifts to right

O since state is processed by columns, this step permutes bytes between the columns

MixColumns

Each column of four bytes is now transformed using a special mathematical function. This function takes as input the four bytes of one column and outputs four completely new bytes, which replace the original column. The result is another new matrix consisting of 16 new bytes. It should be noted that this step is not performed in the last round.

O each column is processed separately

O each byte is replaced by a value dependent on all4 bytes in the column

o effectively a matrix multiplication in GF(2^8) using prime polym(x)

$=x^8+x^4+x^3+x+1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

- **Add Round Key**

The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the

round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

O XOR state with 128-bits of the round key
O again processed by column(though effectively a series of byte operations)
O inverse for decryption is identical since XOR is own inverse, just with Correct round key
O designed to be as simple as possible

- **AES Key Expansion**
  - O takes 128-bit (16-byte) key and expands into array of 44/52/60 32-bit words
  - O start by copying key into first 4 words
  - O then loop creating words that depend on values inprevious&4 places back
    - in 3 of 4 cases just XOR these together
    - every 4th has S-box+ rotate + XOR constant of previous before XOR together
  - o designed to resist known attacks

AES Decryption

Decryption Process

The process of decryption of an AES ciphertext is similar to the encryption process in the reverse order. Each round consists of the four processes conducted in the reverse order −

- Add round key
- Mix columns
- Shift rows
- Byte substitution

Since sub-processes in each round are in reverse manner, unlike for a Feistel Cipher, the encryption and decryption algorithms needs to be separately implemented, although they are very closely related.

O AES decryption is not identical to encryption since steps done in reverse o
but can define an equivalent inverse cipher with steps as for encryption
O but using inverses of each step
O with a different key schedule
O works since result is unchanged when

O swap byte substitution &shift rows
O swap mixcolumns & add (tweaked)round key

Advanced Encryption Standard (AES)

Evaluation Criteria AES Requirements

O private key symmetric block ciphero128-bit data, 128/192/256-bit keys
O stronger & faster than Triple-DES
O active life of 20-30years(+ archival use)
O provide full specification &design details
O both C & Java implementations
o NIST have released all submissions & unclassified analyses

- **AES Evaluation Criteria**
  o initial criteria:
- security– effort to practically cryptanalyse
- cost – computational
- algorithm & implementation characteristics
- final criteria
- general security
- software &hardware implementation ease implementation attacks
- flexibility(in en/decrypt, keying, other factors)

Multiple Encryption and Triple DES

Given the potential vulnerability of DES to a brute-force attack, there has been considerable interest in finding an alternative. One approach is to design a completely new algorithm, of which AES is a prime example.

Another alternative, which would preserve the existing investment in software and equipment, is to use multiple encryption with DES and multiple keys. We begin by examining the simplest example of this second alternative. We then look at the widely accepted triple DES (3DES) approach.

Double DES

The simplest form of multiple encryption has two encryption stages and two keys Given a plaintext P and two encryption keys $K1$ and $K2$, ciphertext C is generated as

$C = E(K2, E(K1, P))$

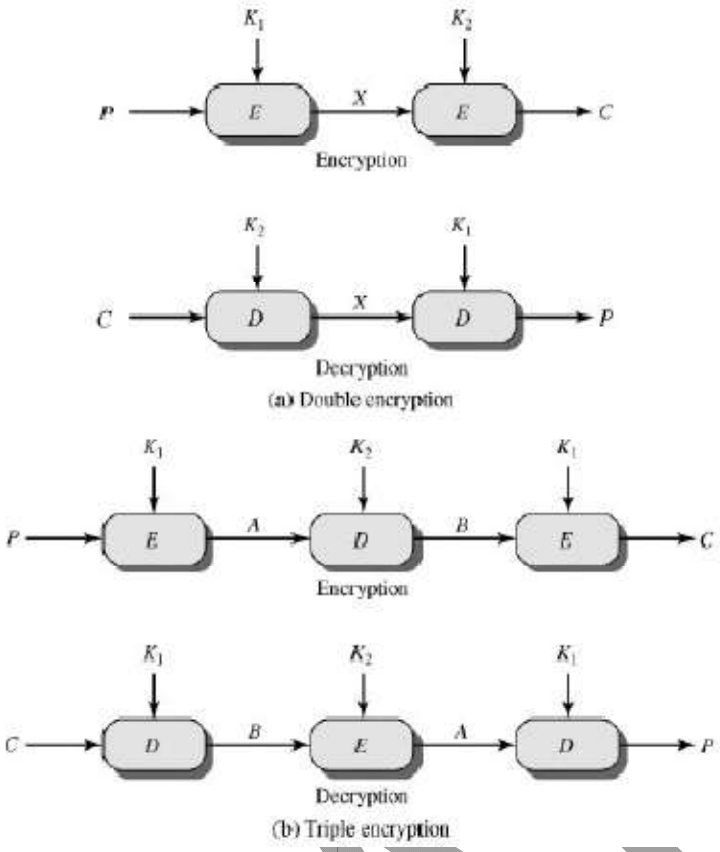


Fig 2.10 (a) Double DES and (b) Triple DES

Multiple Encryption

Decryption requires that the keys be applied in reverse order:

$P = D(K1, D(K2, C))$

For DES, this scheme apparently involves a key length of 56 x 2 = 112 bits, of resulting in a dramatic increase in cryptographic strength. But we need to examine the algorithm more closely.

Meet-in-the-Middle Attack

Thus, the use of double DES results in a mapping that is not equivalent to a single DES encryption. But there is a way to attack this scheme, one that does not depend on any particular property of DES but that will work against any block encryption cipher.

cause discomfort amongst users of DES. However, users did not want to replace DES as it takes an enormous amount of time and money to change encryption algorithms that are widely adopted and

embedded in large security architectures.

The pragmatic approach was not to abandon the DES completely, but to change the manner in which DES is used. This led to the modified schemes of Triple DES (sometimes known as 3DES).

Incidentally, there are two variants of Triple DES known as 3-key Triple DES (3TDES) and 2-key Triple DES (2TDES).

3-KEY Triple DES

Before using 3TDES, user first generate and distribute a 3TDES key K, which consists of three different DES keys K_1, K_2 and K_3. This means that the actual 3TDES key has length $3\times56 = 168$ bits. The encryption scheme is illustrated as follows −

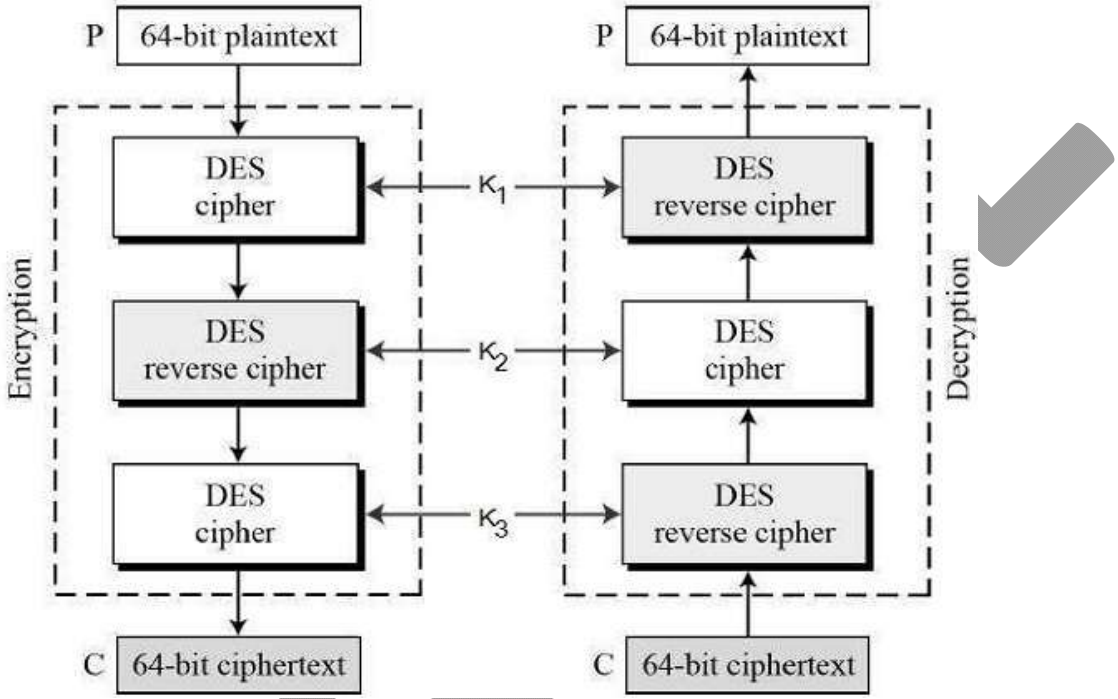


Fig 2.11 Triple DES

The encryption-decryption process is as follows −

- Encrypt the plaintext blocks using single DES with key K_1.
- Now decrypt the output of step 1 using single DES with key K_2.
- Finally, encrypt the output of step 2 using single DES with key K_3.
- The output of step 3 is the ciphertext.

- Decryption of a ciphertext is a reverse process. User first decrypt using K_3, then encrypt with K_2, and finally decrypt with K_1.

Due to this design of Triple DES as an encrypt–decrypt–encrypt process, it is possible to use a 3TDES (hardware) implementation for single DES by setting K_1, K_2, and K_3 to be the same value. This provides backwards compatibility with DES.

Second variant of Triple DES (2TDES) is identical to 3TDES except that $K_3$is replaced by K_1. In other words, user encrypt plaintext blocks with key K_1, then decrypt with key K_2, and finally encrypt with K_1 again. Therefore, 2TDES has a key length of 112 bits.

Triple DES systems are significantly more secure than single DES, but these are clearly a much slower process than encryption using single DES.

TripleDES

Clear a replacement for DES was needed theoretical attacks that can break it demonstrated exhaustive key search attacks AES is a new cipher alternative prior to this alternative was to use multiple encryption with DES implementations Triple-DES is the chosen form.

Why Triple-DES?

NOT same as some other single-DES use, but have meet-in-the-middle attack works whenever use a cipher twice since $X = E_{K1}[P] = D_{K2}[C]$ attack by encrypting P with all keys and store then decrypt C with keys and match X value can show takes $O(2^{56})$steps

Triple-DES with Two-Keys

- O hence must use 3 encryptions
  - would seem to need 3 distinct keys
- O but can use 2 keys with E-D-E sequence
  - $C = E_{K1}[D_{K2}[E_{K1}[P]]]$
  - nb encrypt & decrypt equivalent in security
  - ifK1=K2 then can work with single DES
- O standardized in ANSIX9.17 &ISO8732
- O no current known practical attacks

Triple-DES with Three-Keys

- O although are no practical attacks on two-key Triple-DES have some indications
- O can use Triple-DES with Three-Keys to avoid even these
  - $C = E_{K3}[D_{K2}[E_{K1}[P]]]$
- O has been adopted by some Internet applications, eg PGP, S/MIME

Placement of Encryption Function

- can place encryption function at various layers in OSI Reference Model
  link encryption occurs at layers 1 or 2
    - end-to-end can occur at layers 3, 4, 6, 7
    - as move higher less information is encrypted but it is more secure though more complex with more entities and keys

Traffic Confidentiality

- Is monitoring of communications flows between parties
  useful both in military & commercial spheres
  can also be used to create a covert channel
- link encryption obscures header details
  but overall traffic volumes in networks and at end-points is still visible
- traffic padding can further obscure flows
  but at cost of continuous traffic

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, four "modes of operation" have been defined by NIST (FIPS 81). 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 four modes are intended to cover virtually all the possible applications of encryption for which a block cipher could be used.

Mode	Description	Typical Application
Electronic Codebook (ECB)	Each block of 64 plaintext bits is encoded independently using the same key.	• Secure transmission of single values (e. g., an encryption key)
Cipher Block Chaining (CBC)	The input to the encryption algorithm is the XOR of the next 64 bits of plaintext and the preceding 64 bits of ciphertext.	• General-purpose block-oriented transmission • Authentication
Cipher Feedback (CFB)	Input is processed *j* bits at a time. Preceding ciphertext is used as input to the encryption algorithm to produce pseudorandom output, which is XORed with plaintext to produce next unit of ciphertext.	• General-purpose stream-oriented transmission • Authentication
Output Feedback (OFB)	Similar to CFB, except that the input to the encryption algorithm is the preceding DES output.	• Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted counter. The counter is incremented for each subsequent block.	• General-purpose block-oriented transmission • Useful for high-speed requirements

Electronic Codebook Book (ECB)

Operation

- The user takes the first block of plaintext and encrypts it with the key to produce the first block of ciphertext.
- He then takes the second block of plaintext and follows the same process with same key and so on so forth.

The ECB mode is **deterministic**, that is, if plaintext block P1, P2,…, Pm are encrypted twice under the same key, the output ciphertext blocks will be the same.

In fact, for a given key technically we can create a codebook of ciphertexts for all possible plaintext blocks. Encryption would then entail only looking up for required plaintext and select the corresponding ciphertext. Thus, the operation is analogous to the assignment of code words in a codebook, and hence gets an official name − Electronic Codebook mode of operation (ECB).

O message is broken into independent blocks which are encrypted
O each block is a value which is substituted, like a codebook, hence name
O each block is encoded independently of the other blocks

$C_i = DES_{K1}(P_i)$

o uses: secure transmission of single values

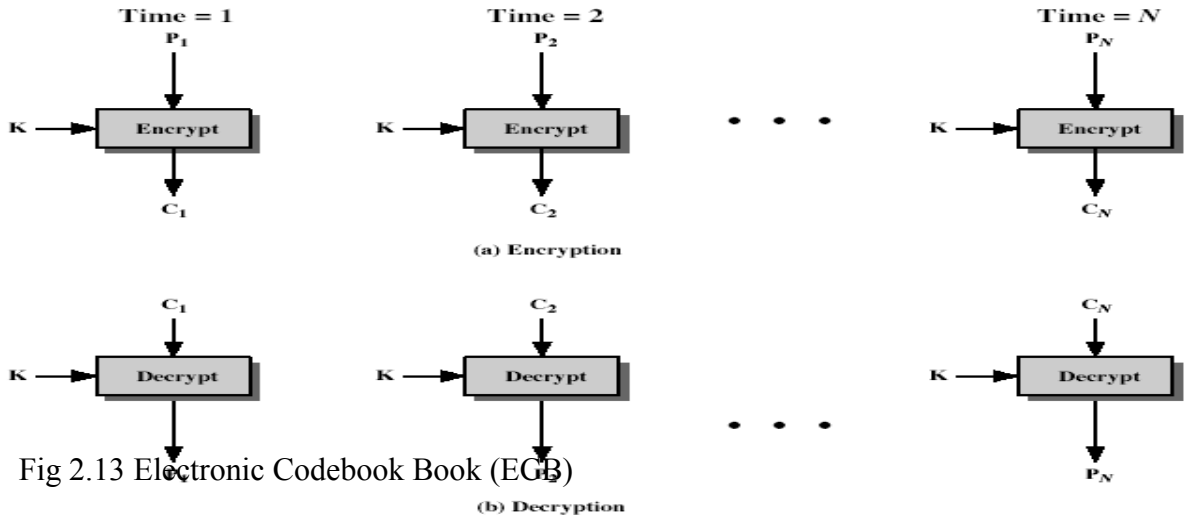


Fig 2.13 Electronic Codebook Book (ECB)

- **Advantages and Limitations of ECB**
  - O repetitions in message may show in ciphertext
  - O if aligned with message block
  - O particularly with data such graphics
  - O or with messages that change very little, which become a code-book analysis problem
  - o weakness due to encrypted message blocks being independent
  - o main use is sending a few blocks of data

Cipher Block Chaining (CBC)

- O message is broken into blocks
- O but these are linked together in the encryption operation
- O each previous cipher blocks is chained with current plaintext block, hence name
- O use Initial Vector(IV) to start process

$$C_i = DES_{K1}(P_i XOR C_{i-1})$$

$C_{-1} = IV$

- O uses: bulk data encryption, authentication

CBC mode of operation provides message dependence for generating ciphertext and makes the system non-deterministic.

Operation

The operation of CBC mode is depicted in the following illustration. The steps are as follows −

- Load the n-bit Initialization Vector (IV) in the top register.
- XOR the n-bit plaintext block with data value in top register.
- Encrypt the result of XOR operation with underlying block cipher with key K.
- Feed ciphertext block into top register and continue the operation till all plaintext blocks are processed.
- For decryption, IV data is XORed with first ciphertext block decrypted. The first ciphertext block is also fed into to register replacing IV for decrypting next ciphertext block.

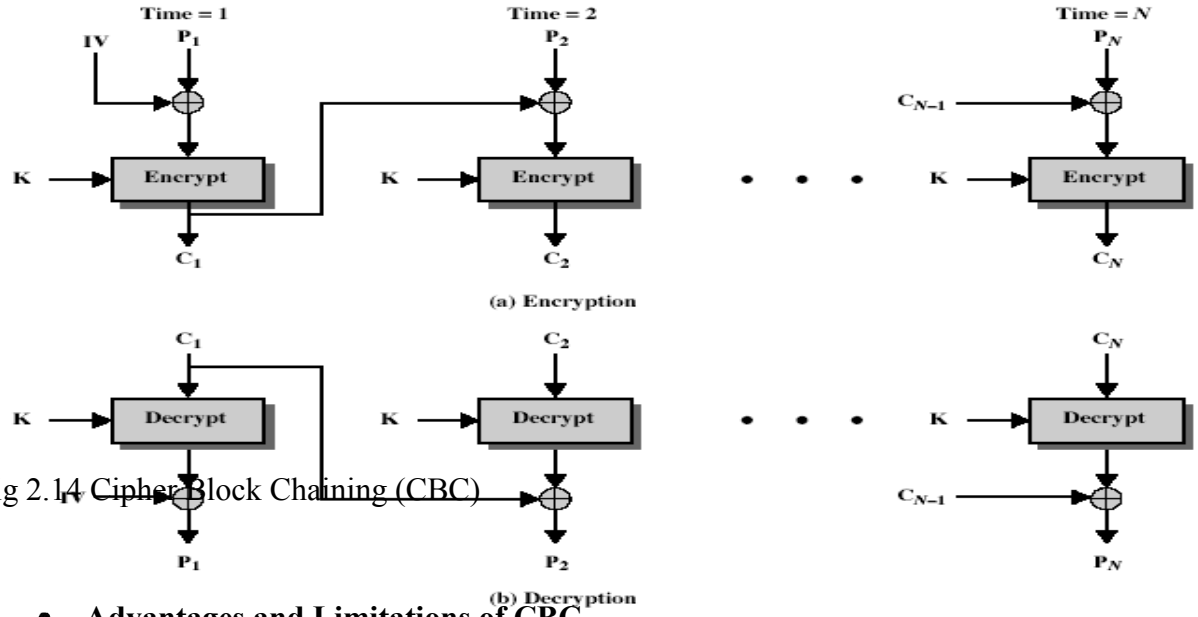


Fig 2.14 Cipher Block Chaining (CBC)

- **Advantages and Limitations of CBC**
  - O each ciphertext block depends on all message blocks
  - O thus a change in the message affects all ciphertext blocks after the change as well as the original block
  - O need Initial Value(IV)known to sender & receiver
    - however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate
    - hence either IV must be a fixed value (as in EFTPOS) or it must be sent encrypted in ECB mode before rest of message
  - O at end of message, handle possible last short block
    - by padding either with known non-data value (eg nulls)
    - or pad last block with count of pad size
      - eg. [b1 b2 b3 0 0 0 0 5]<- 3 data bytes, then 5 bytes pad + count

Cipher Feed Back(CFB)

O message is treated as a stream of bits
O added to the output of the block cipher
O result is feedback for next stage (hence name)
O standard allows any number of bit(1,8 or64 or whatever) to be feedback
O denoted CFB-1, CFB-8, CFB-64etc
O is most efficient to use all 64 bits(CFB-64)

$$C_i = P_i \text{ XOR } DES_{K1}(C_{i-1})$$

C_{-1}=IV

O uses: stream data encryption, authentication

In this mode, each ciphertext block gets 'fed back' into the encryption process in order to encrypt the next plaintext block.

Operation

The operation of CFB mode is depicted in the following illustration. For example, in the present system, a message block has a size 's' bits where 1 < s < n. The CFB mode requires an initialization vector (IV) as the initial random n-bit input block. The IV need not be secret. Steps of operation are −

- Load the IV in the top register.
- Encrypt the data value in top register with underlying block cipher with key K.
- Take only 's' number of most significant bits (left bits) of output of encryption process and XOR them with 's' bit plaintext message block to generate ciphertext block.
- Feed ciphertext block into top register by shifting already present data to the left and continue the operation till all plaintext blocks are processed.
- Essentially, the previous ciphertext block is encrypted with the key, and then the result is XORed to the current plaintext block.
- Similar steps are followed for decryption. Pre-decided IV is initially loaded at the start of decryption.

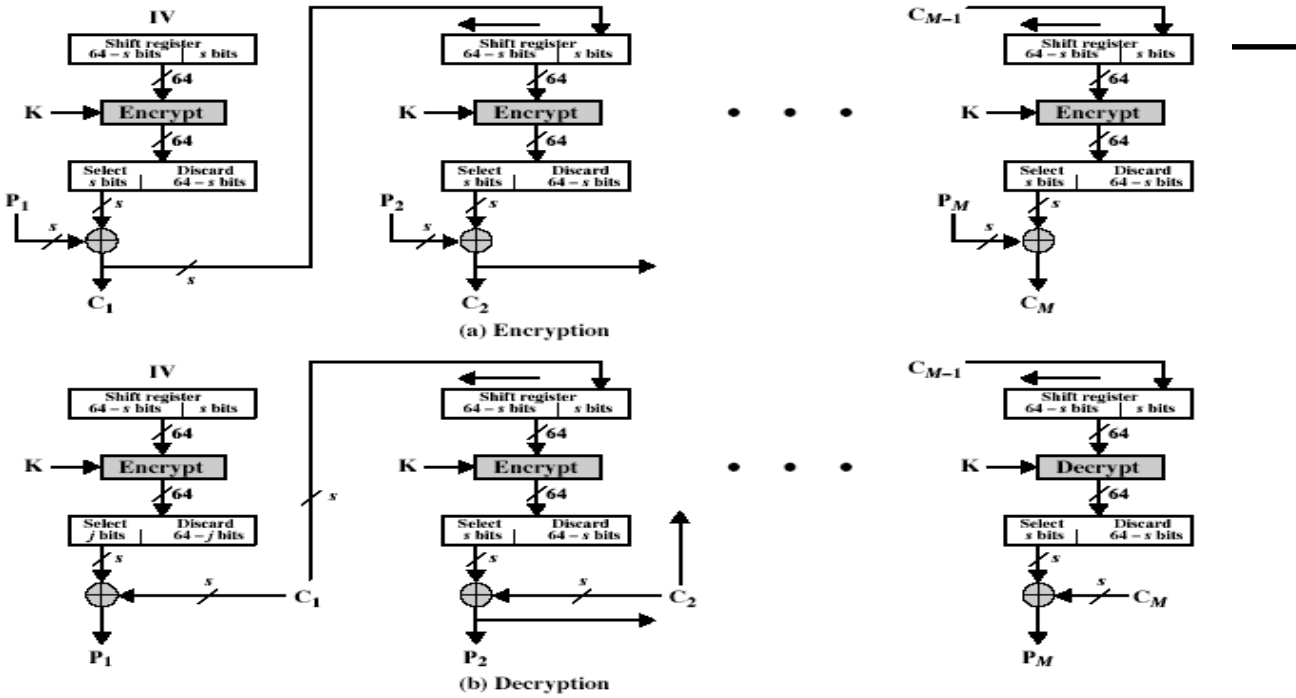


Fig 2.15 Cipher Feed Back(CFB)

- **Advantages and Limitations of CFB**
  - O appropriate when data arrives in bits/bytes
  - O most common stream mode
  - O limitation is need to stall while do block encryption after every n-bits
  - O note that the block cipher is used in encryption mode at both ends
  - O errors propagate for several blocks after the error

Output Feed Back (OFB)

- O message is treated as a stream of bits
- O output of cipher is added to message
- O output is then feedback (hencename)
- O feedback is independent of message
- O can be computed in advance

$C_i = P_i XOR\ _iO_i =$

$DES_{K1}(O_{i-1})O-$

$_1 = IV$

- O uses: stream encryption over noisy channels

It involves feeding the successive output blocks from the underlying block cipher back to it. These feedback blocks provide string of bits to feed the encryption algorithm which act as the key-stream generator as in case of CFB mode.

The key stream generated is XOR-ed with the plaintext blocks. The OFB mode requires an IV as the initial random n-bit input block. The IV need not be secret.

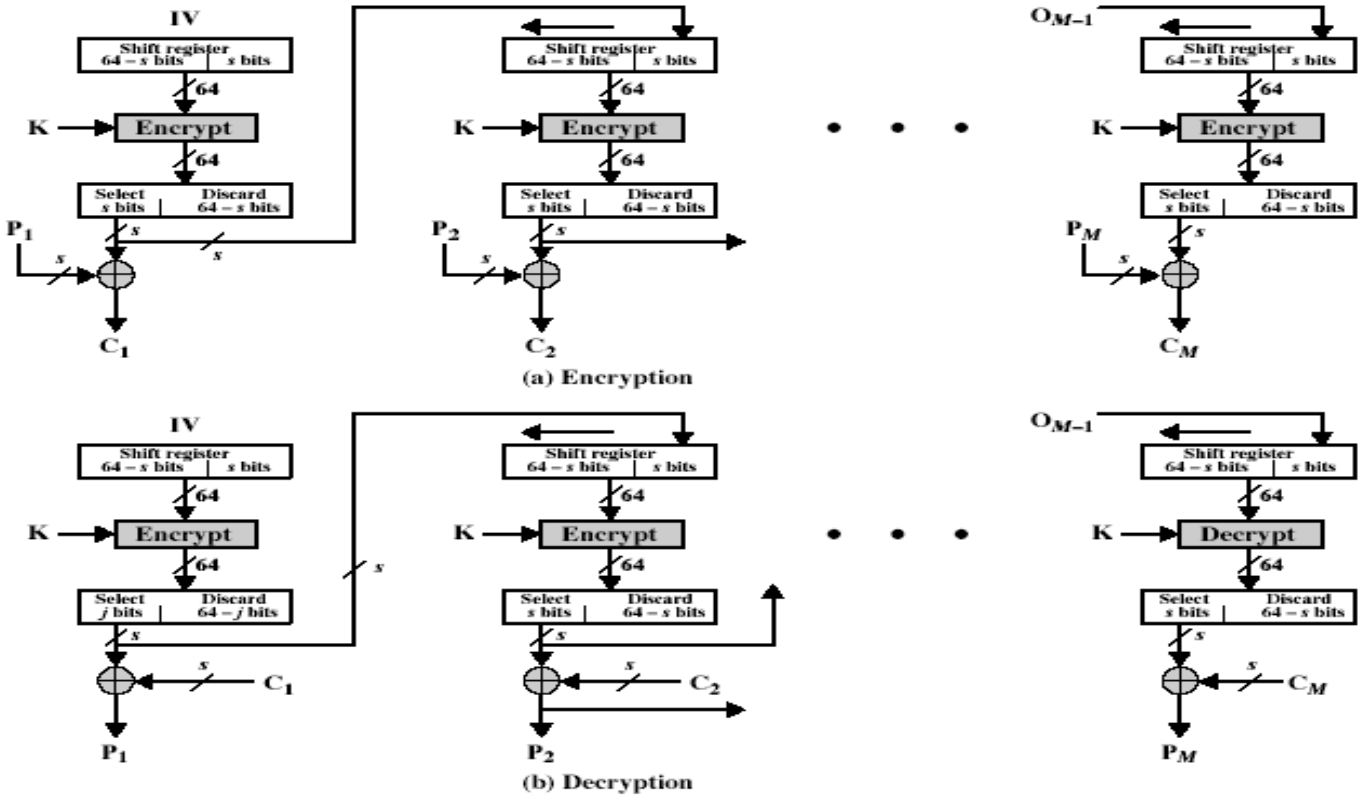


Fig 2.16 Output Feed Back (OFB)

- **Advantages and Limitations of OFB**
  - O used when error feedback a problem or where need to encryptions before message is available
  - O superficially similar to CFB
  - O but feedback is from the output of cipher and is independent of message o a variation of a Vernam cipher
  - O hence must never reuse the same sequence(key+IV)
  - O sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs
  - O originally specified with m-bit feedback in the standards
  - O subsequent research has shown that only **OFB-64** should ever be used

Counter (CTR) Mode

It can be considered as a counter-based version of CFB mode without the feedback. In this mode, both the sender and receiver need to access to a reliable counter, which computes a new shared value each time a ciphertext block is exchanged. This shared counter is not necessarily a secret value, but challenge is that both sides must keep the counter synchronized.

Operation

Both encryption and decryption in CTR mode are depicted in the following illustration. Steps in operation are −

- Load the initial counter value in the top register is the same for both the sender and the receiver. It plays the same role as the IV in CFB (and CBC) mode.
- Encrypt the contents of the counter with the key and place the result in the bottom register.
- Take the first plaintext block P1 and XOR this to the contents of the bottom register. The result of this is C1. Send C1 to the receiver and update the counter. The counter update replaces the ciphertext feedback in CFB mode.
- Continue in this manner until the last plaintext block has been encrypted.
- The decryption is the reverse process. The ciphertext block is XORed with the output of encrypted contents of counter value. After decryption of each ciphertext block counter is updated as in case of encryption.

O a "new" mode, though proposed early on
O similar to OFB but encrypts counter value rather than any feedback value
O must have a different key & counter value for every plaintext block (never reused)

$C_i = P_i \text{XOR } O_i$

$O_i = DES_{K1}(i)$

O uses: high-speed network encryptions

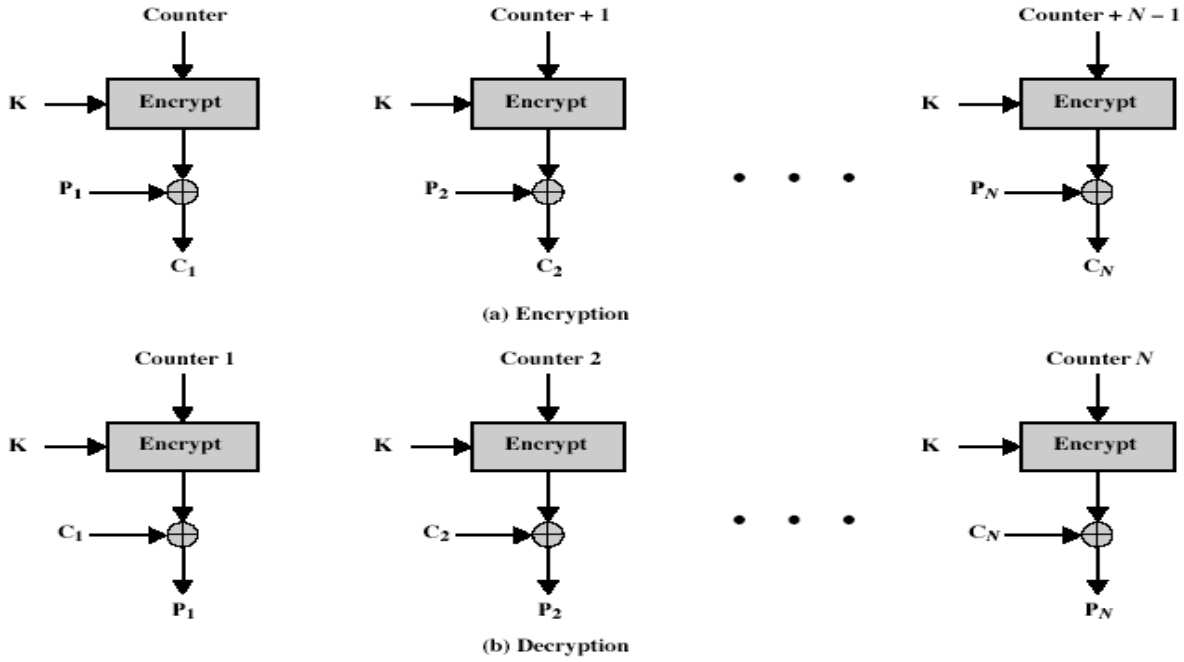


Fig 2.17. Counter (CTR) Mode

Advantages and Limitations of CTR

O efficiency

- can do parallel encryptions
- in advance of need

. good for bursty high speed links

O random access to encrypted data blocks

O provable security (good as other modes)

O but must ensure never reuse key/counter values, otherwise could break (cf OFB)

Stream Ciphers and RC4

In this section we look at perhaps the most popular symmetric stream cipher, RC4. We begin with an overview of stream cipher structure, and then examine RC4.

Stream Cipher Structure

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 is a 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. For now, we simply say that a pseudorandom stream is one that is unpredictable without knowledge of the input key.

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.

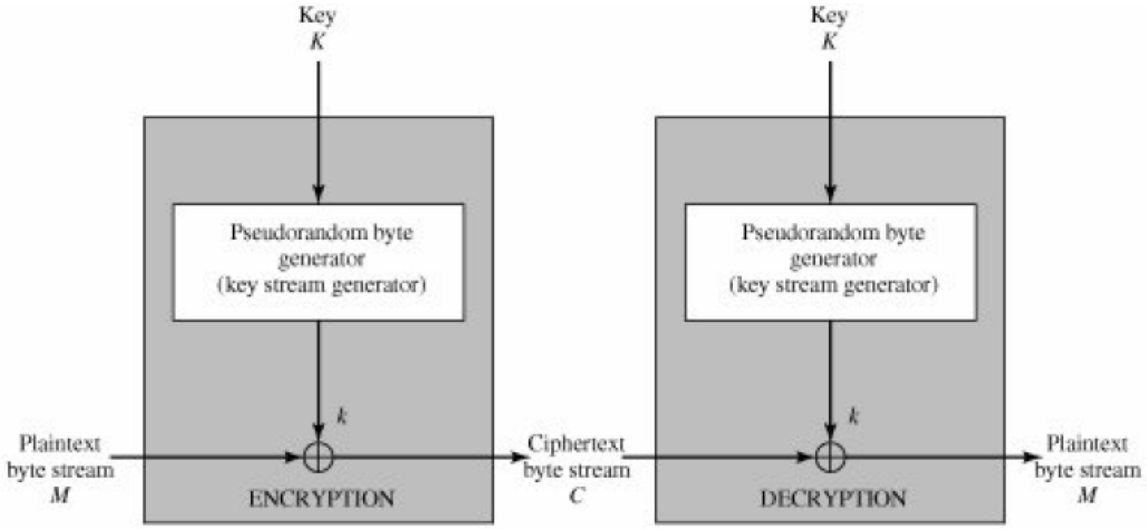
For example, if the next byte generated by the generator is 01101100 and the next plaintext byte is 11001100, then the resulting ciphertext byte is

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

Decryption requires the use of the same pseudorandom sequence:

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

Stream Cipher Diagram



The stream cipher is similar to the one-time pad. The difference is that a onetime pad uses a genuine random number stream, whereas a stream cipher uses a pseudorandom number stream.
The following 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. 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 as apply for 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 block cipher of comparable key length.

The primary advantage of a stream cipher is that stream ciphers are almost always faster and use far less code than do block ciphers.

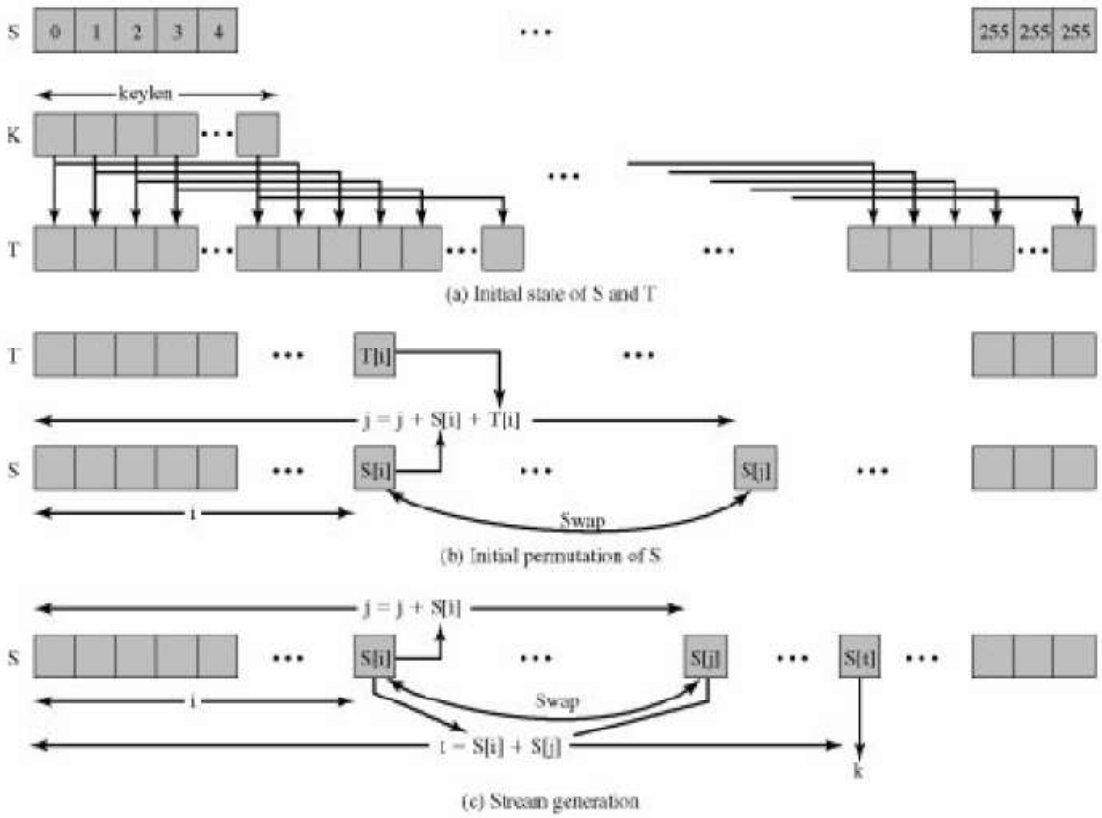
The RC4 Algorithm

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.

Analysis shows that the period of the cipher is overwhelmingly likely to be greater than 10100 [ROBS95a]. Eight to sixteen machine operations are required per output byte, and the cipher can be expected to run very quickly in software. RC4 is used in the SSL/TLS (Secure Sockets Layer/Transport Layer Security) standards that have been defined for communication between Web browsers and servers.

It is also used in the WEP (Wired Equivalent Privacy) protocol and the newer WiFi Protected Access (WPA) protocol that are part of the IEEE 802.11 wireless LAN standard. RC4 was kept as a trade secret by RSA Security. In September 1994, the RC4 algorithm was anonymously posted on the Internet on the Cypherpunks anonymous remailers list.

The RC4 algorithm is remarkably simply and quite easy to explain. A variable-length 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],..., 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* (see Figure 6.8) 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.



Modular Arithmetic

Given any positive integer *n* and any nonnegative integer *a*, if we divide *a* by *n*, we get an integer quotient *q* and an integer remainder *r* that obey the following relationship:

$$a = qn + r \qquad 0 \le r < n; q = \lfloor a/n \rfloor$$

where $\lfloor x \rfloor$ is the largest integer less than or equal to *x*.

$a = 11;$	$n = 7;$	$11 = 1 \times 7 + 4;$	$r = 4$	$q = 1$
$a = -11;$	$n = 7;$	$-11 = (-2) \times 7 + 3;$	$r = 3$	$q = -2$

If *a* is an integer and *n* is a positive integer, we define *a* mod *n* to be the remainder when *a* is divided by
n. The integer *n* is called the **modulus**. Thus, for any integer *a*, we can always write:

$$a = \lfloor a/n \rfloor \times n + (a \bmod n)$$

11 mod 7 = 4;	-11 mod 7 = 3

Modular Arithmetic Operations

Note that, by definition (Figure 4.2), the (mod *n*) operator maps all integers into the set of integers {0,1,... (*n* 1)}. This suggests the question: Can we perform arithmetic operations within the confines of this
set? It turns out that we can; this technique is known as **modular arithmetic**.

Modular arithmetic exhibits the following properties:

1. [(*a* mod *n*) + (*b* mod *n*)] mod *n* = (*a* + *b*) mod *n*

2. [(*a* mod *n*) (*b* mod *n*)] mod *n* = (*a b*) mod *n*

3. [(*a* mod *n*) x (*b* mod *n*)] mod *n* = (*a* x *b*) mod *n*

The Euclidean Algorithm

One of the basic techniques of number theory is the Euclidean algorithm, which is a simple procedure for determining the greatest common divisor of two positive integers.

Greatest Common Divisor

Recall that nonzero *b* is defined to be a divisor of *a* if *a* = *mb* for some *m*, where *a*, *b*, and *m* are integers. We will use the notation gcd(*a*, *b*) to mean the **greatest common divisor** of *a* and *b*. The positive integer *c* is said to be the greatest common divisor of *a* and *b* if

1. *c* is a divisor of *a* and of *b*;

2. any divisor of *a* and *b* is a divisor of *c*.

An equivalent definition is the following:

gcd(*a*, *b*) = max[*k*, such that *k*|*a* and *k*|*b*]

Because we require that the greatest common divisor be positive, gcd(*a*, *b*) = gcd(*a*, *b*) = gcd(*a*, *b*) =gcd(*a*, *b*). In general, gcd(*a*, *b*) = gcd(|*a*|, |*b*|).

gcd(60, 24) = gcd(60, 24) = 12

Also, because all nonzero integers divide 0, we have gcd(*a*, 0) = |*a*|.

We stated that two integers *a* and *b* are relatively prime if their only common positive integer factor is 1.

This is equivalent to saying that *a* and *b* are relatively prime if gcd(*a*, *b*) = 1.

8 and 15 are relatively prime because the positive divisors of 8 are 1, 2, 4, and 8, and the positive divisors of 15 are 1, 3, 5, and 15, so 1 is the only integer on both lists.

Finding the Greatest Common Divisor

The Euclidean algorithm is based on the following theorem: For any nonnegative integer *a* and any positive integer *b*,

$$\gcd(a, b) = \gcd(b, a \bmod b)$$

gcd(55, 22) = gcd(22, 55 mod 22) = gcd(22, 11) = 11

POSSIBLE QUESTIONS

PART A (20 x 1 = 20 marks)
(ONLINE EXAMINATION)

PART B (5 x 6 = 30 marks)

1. Describe the block modes of operations of DES with their advantages.
2. Explain with neat diagram AES encryption and decryption algorithm.
3. Describe in detail about Block cipher modes of operation.
4. Describe in detail about multiple encryption and triple DES

PART – C (1 x 10 = 10 marks)

1. Explain with neat diagram AES encryption and decryption algorithm.
2. Describe the block modes of operations of DES with their advantages.
3. Describe in detail about Block cipher modes of operation.

KARPAGAM ACADEMY OF HIGHER EDUCATION						
DEPARTMENT OF COMPUTER SCIENCE						
I M.Sc CS						
CRYPTOGRAPHY AND NETWORK SECURITY						
	UNIT 2					
S.NO	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	A _____ is one which a block of plain text is treated as a whole and used to produce a ciphertext block of equal length.	stream cipher	symmetric cipher	block cipher	none	block cipher
2	The _____ has been the most widely used encryption algorithm until recently	DES	AES	AFS	DEM	DES
3	A _____ is one that encrypts a digital data stream one bit or one byte at a time.	block cipher	symmetric cipher	stream cipher	plain	stream cipher
4	A block cipher operates on a plain text block of n bits to produce a ciphertext block of bits	n*n*	n*n	n power n	n	n
5	most symmetric block encryption algorithms in current use are based on a structure referred to as a block cipher.	Feistel	block	stream	none	Feistel
6	Feistel refers to this as the _____ cipher because it allows for the maximum number of possible encryption mappings from the	asymmetric	product	symmetric	ideal block	ideal block
7	the terms _____ and _____ were introduced by claude shannon to capture the two basic building blocks for any	diffusion confusion	diffusion	permutation	none	diffusion confusion
8	In _____ the statistical structure of the plain text is dissipated into long range statistics of the ciphertext.	diffusion	substitution	permutation	confusion	diffusion
9	_____ seeks to make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as	permutation	substitution	confusion	diffusion	confussio n
10	A_____ is performed on the left half of the data	substitution	permutation	confusion	transformation	substitutio n
11	A_____ is performed that consists of the interchange of the two halves of the data	substitution	permutation	confusion	transformation	permutatio n
12	A block size of _____ bits has been considered a reasonable tradeoff and was nearly universal in block cipher design	16	8	32	64	64
13	_____ key size means greater security but may decrease encryption/decryption speed	larger	smaller	medium	double	larger
14	The DES adopted in _____ by the national bureau of standards	1977	1978	1979	1980	1977
15	NIST stands for	national institute of standards and	national institute of state and	national industrial of standards and	national institute of state and	national institute of
16	For DES data are encrypted in ___ blocks using a _____bit key.	8 bit,25	32bit,45	64 bit,56	16 bit,55	64 bit,56

17	The left and right halves of the output are swapped to produce the	preoutput	preinput	postinput	postoutput	preoutput
18	The _____ function is the same for each round but a different subkey is produced because of the repeated shifts of the key bits.	substitution	permutation	confusion	transformation	permutation
19	with a key length of 56 bits there are 2power 56 possible keys which is approximately keys	7.2*10 power 16	7.5*10 power 16	7.7*10 power 16	795*10 power 16	7.2*10 power 16
20	EFF stands for	Electronic Frontier	Electronic Form Foundation	Electric Form Foundation	Email Frontier Foundation	Electronic Frontier
21	AES stands for	Aavent Encryption	Advanced Enable Standard	Advanced Encryption Standard	Advanced Encryption Stante	Advanced Encryption
22	How many number of rounds are used in AES if key size is 24 bytes	10	14	12	16	12
23	_____ refers to the effort required to cryptanalyze an algorithm.	security	cost	randomness	product	security
24	_____ of the mathematical basis for the algorithms security	soundness	randomness	cost	product	soundness
25	A candidate algorithm shall be judged according to relative of design.	permutation	complexity	random	simplicity	simplicity
26	_____ algorithms with greater flexibility will meet the needs of more user than less flexible ones.	candidate	primary	super	foreign	candidate
27	_____ refers to the ability to change keys quickly and with a minimum of resources.	key agility	key enable	key distribution	key change	key agility
28	The forward substitute byte transformation called _____ bytes	max	super	min	sub	sub
29	The _____ key expansion algorithm takes as input a 4 word key and produce a linear of 44 words	AES	DES	CIS	AEM	AES
30	AES can be implemented very efficiently on an _-bit processor	8	16	32	64	8
31	_____ uses an s-box to perform a byte-by-byte substitution of the block	substitution bytes	permutation butes	diffusion butes	confussion butes	substitution bytes
32	ECB stands for	electronic coin book	electronic code book	electronic code bill	email code book	electronic code book
33	CBC stands for	Cipher Block Chating	Cylinder Block Chaining	Cipher Base Chaining	Cipher Block Chaining	Cipher Block
34	CFB stands for _____	Cipher Formback	Cylinder Feedback	Cipher Front	Cipher Feedback	Cipher Feedback
35	The ____mode is similar in structure to that of CFB	OFB	CTR	CTL	CTTL	OFB
36	the _____ mode has increased recently with applications to ATM network security	CTL	OFB	CTR	CTTL	CTR

37	ATM stands for	Asynchronous transfer mode	Asynchronous teller mode	All time teller mode	Asynchronous target mode	Asynchron ous
38	_____ the execution of the underlying encryption algorithm does not depend on input of the plaintext or ciphertext.	preprocessing	software	hardware	none	preproces sing
39	_____ can be shown that CRT is at least as secure as the other modes	random access	Provable Security	simplicity	software	Provable Security
40	_____CRT modes requires only the implementation of the encryption algorithm and not decryption algorithm.	simplicity	random access	preprocessing	software	simplicity
41	How many number of rounds are used in DES	8	16	10	12	16
42	_____ is a stream cipher designed in 1987 by Ron Rivest for RSA security	RC4	RC5	RC6	RC7	RC4
43	The round function F in DES is applied to the _____ part of the data	left	right	alternatively	both	right
44	A small change in either plain text or the key should produce a significant change in ciphertext is called	permutation	confusion	avalanche effect	diffusion	avalanche effect
45	_____ algorithm is based on principle that the greatest common divisor of two numbers	RC4	Euclidean	AES	DES	Euclidean
46	_____ is a stream cipher designed in 1987 by Ron Rivest for RSA security	RC4	RC5	RC6	RC7	RC4
47	_____ encryption has two encryption stages and two keys.	Two DES	Double DES	Triple DES	DES	Double DES
48	In which block cipher modes of operation, blocks are enciphered independently of other blocks	CBC	CFC	OFB	ECB	ECB
49	_____ is used in SSL/TLS standards that have been defined for communication between web browser and servers.	TRNG	PRNG	stream generation	RC4	RC4
50	In AES, the cipher begins and ends with an _____ stage	Sub bytes	Shift rows	Mix columns	AddRoundKey	AddRound Key
51	Which of the following operations is required to convert 32bit right half into 48bit for the DES cipher.	permutation	substitution/choice	Expansion/Permutati on	None	Expansion /Permutati
52	In AES cipher, which stage is omitted from the final round	Sub bytes	Shift rows	Mix columns	AddRoundKey	Mix columns
53	Which of the following is a valid block length for the AES cipher	128 bits	192 bits	256 bits	All the above	128 bits
54	_____ , one bit of plain text is encrypted at a time.	stream cipher	block cipher	Both (a) and (b)	none	stream cipher
55	_____ substitution is a process that accepts 48 bits from XOR operation	S- box	P- box	Expansion/Permutati on	Key transformation	S- box
56	If the key size is 24 bytes in AES, what would be the number of rounds	10	12	15	14	12

57	_____ is a round cipher based on the Rijndael algorithm that uses a 128-bit block of data.	DES	AES	AER	DER	AES
58	What result the MOD function will produce MOD(30,7)?	3	0	2	1	2
59	Two integers a and b are said to be _____ modulo n, if (a mod n) = (b mod n).	modular	congruent	arithmetic	none	congruent
60	The actual algorithm in the AES encryption schema is _____	Blowfish	IDEA	Rijndael	RC4	Rijndael

UNIT-III
Syllabus

Confidentiality using Symmetric Encryption – Placement of Encryption Function – Traffic Confidentiality – Key Distribution – Public key Cryptography and RSA – Principles of Public Key Cryptosystems – The RSA Algorithm- Basic prime numbers and Discrete Logarithms -Key Management – Diffie Hellman Key Exchange.

CONFIDENTIALITY USING SYMMETRIC ENCRYPTION

Placement of Encryption Function

If encryption is to be used to counter attacks on confidentiality, we need to decide what to encrypt and where the encryption function should be located. To begin, this section examines the potential locations of security attacks and then looks at the two major approaches to encryption placement: link and end to end.

Potential Locations for Confidentiality Attacks

As an example, consider a user workstation in a typical business organization. Figure suggests the types of communications facilities that might be employed by such a workstation and therefore gives an indication of the points of vulnerability.

Points of Vulnerability

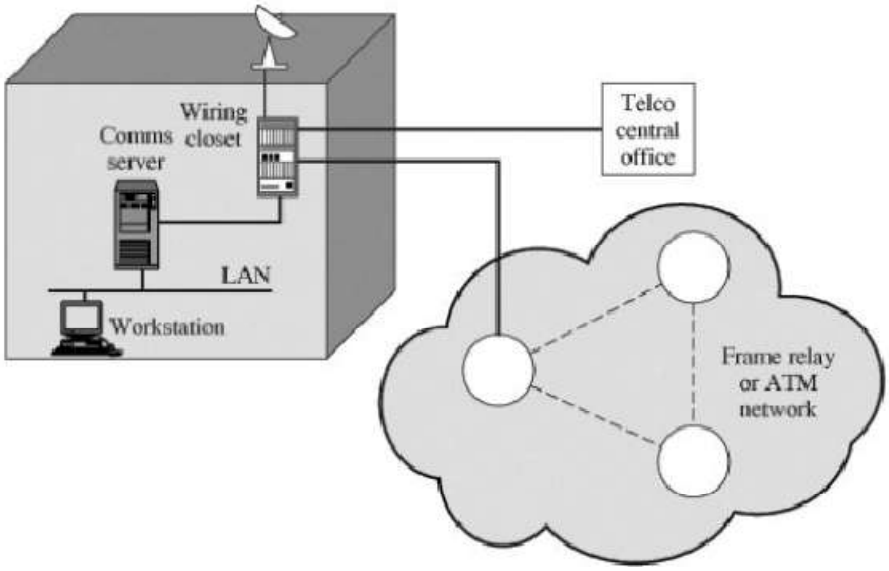


Fig 3.1 Points of Vulnerability

In most organizations, workstations are attached to local area networks (LANs). Typically, the user can reach other workstations, hosts, and servers directly on the LAN or on other LANs in the same building that are interconnected with bridges and routers. Here, then, is the first point of vulnerability. In this case, the main concern is eavesdropping by another employee.

Typically, a LAN is a broadcast network: Transmission from any station to any other station is visible on the LAN medium to all stations. Data are transmitted in the form of frames, with each frame containing the source and destination address. An eavesdropper can monitor the traffic on the LAN and capture any traffic desired on the basis of source and destination addresses. If part or all of the LAN is wireless, then the potential for eavesdropping is greater.

Furthermore, the eavesdropper need not necessarily be an employee in the building. If the LAN, through a communications server or one of the hosts on the LAN, offers a dial-in capability, then it is possible for an intruder to gain access to the LAN and monitor traffic. Access to the outside world from the LAN is almost always available in the form of a router that connects to the Internet, a bank of dial-out modems, or some other type of communications server. From the communications server, there is a line leading to a wiring closet.

The wiring closet serves as a patch panel for interconnecting internal data and phone lines and for providing a staging point for external communications. The wiring closet itself is vulnerable. If an intruder can penetrate to the closet, he or she can tap into each wire to determine which are used for data transmission. After isolating one or more lines, the intruder can attach a low-power radio transmitter. The resulting signals can be picked up from a nearby location (e.g., a parked van or a nearby building).

In addition to the potential vulnerability of the various communications links, the various processors along the path are themselves subject to attack. An attack can take the form of attempts to modify the hardware or software, to gain access to the memory of the processor, or to monitor the electromagnetic emanations. These attacks are less likely than those involving communications links but are nevertheless a source of risk.

Thus, there are a large number of locations at which an attack can occur. Furthermore, for wide area communications, many of these locations are not under the physical control of the end user. Even in the case of local area networks, in which physical security measures are possible, there is always the threat of the disgruntled employee.

Confidentiality using Symmetric Encryption have two major placement alternatives

- **link encryption**
- encryption occurs independently on every link
- must decrypt traffic between links
- requires many devices, but paired keys

• **end-to-end encryption**
• encryption occurs between original source and final destination
• need devices at each end with shared keys

Placement of Encryption-logical placement
• can place encryption function at various layers in OSI
Reference Model
• link encryption occurs at layers 1 or 2
• end-to-end can occur at layers 3, 4, 6, 7
• as move higher less information is encrypted but it is more secure though more complex with more entities and keys
End-End Logical Placement

Traffic Analysis
When using end-to-end encryption must leave headers in clear
• so network can correctly route information
• So, contents protected, but traffic pattern flows are not
• ideally want both at once
• end-to-end protects data contents over entire path and provides authentication
• link protects traffic flows from monitoring

• is monitoring of communications flows between parties
• useful both in military & commercial spheres
• can also be used to create a covert channel
• link encryption obscures header details
• but overall traffic volumes in networks and at end-points is still visible
• traffic padding can further obscure flows
• but at cost of continuous traffic

Link versus End-to-End Encryption
The most powerful and most common approach to securing the points of vulnerability highlighted in the preceding section is encryption. If encryption is to be used to counter these attacks, then we need to decide what to encrypt and where the encryption gear should be located. There are two fundamental alternatives: link encryption and end-to-end encryption.

Placement of Encryption-Basic Approach

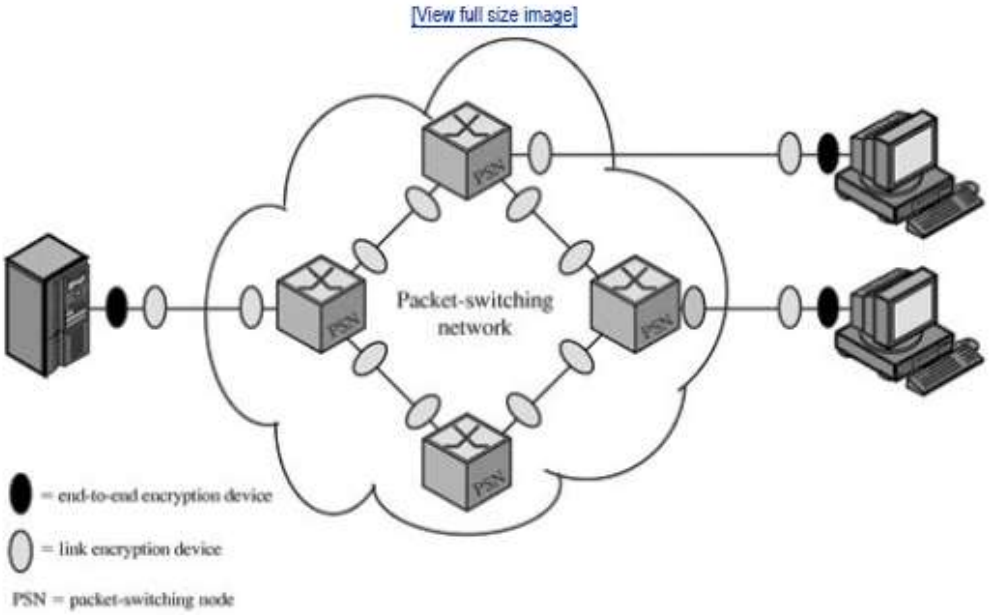


Fig 3.2 Encryption Across a Packet-Switching Network

With link encryption, each vulnerable communications link is equipped on both ends with an encryption device. Thus, all traffic over all communications links is secured. Although this recourse requires a lot of encryption devices in a large network, its value is clear. One of its disadvantages is that the message must be decrypted each time it enters a switch (such as a frame relay switch) because the switch must read the address (logical connection number) in the packet header in order to route the frame. Thus, the message is vulnerable at each switch.

If working with a public network, the user has no control over the security of the nodes. Several implications of link encryption should be noted. For this strategy to be effective, all the potential links in a path from source to destination must use link encryption. Each pair of nodes that share a link should share a unique key, with a different key used on each link. Thus, many keys must be provided with end-to-end encryption, the encryption process is carried out at the two end systems. The source host or terminal encrypts the data. The data in encrypted form are then transmitted unaltered across the network to the destination terminal or host. The destination shares a key with the source and so is able to decrypt the data. This plan seems to secure the transmission against attacks on the network links or switches.

Thus, end-to-end encryption relieves the end user of concerns about the degree of security of networks and links that support the communication. There is, however, still a weak spot. Consider the following situation. A host connects to a frame relay or ATM network, sets up a logical connection to another host, and is prepared to transfer data to that other host by using end-to-end encryption. Data are transmitted over such a network in the form of packets that consist of a header and some user data.

What part of each packet will the host encrypt? Suppose that the host encrypts the entire packet, including the header. This will not work because, remember, only the other host can perform the decryption. The frame relay or ATM switch will receive an encrypted packet and be unable to read the header. Therefore, it will not be able to route the packet. It follows that the host may encrypt only the user data portion of the packet and must leave the header in the clear. Thus, with end-to-end encryption, the user data are secure. However, the traffic pattern is not, because packet headers are transmitted in the clear.

On the other hand, end-to-end encryption does provide a degree of authentication. If two end systems share an encryption key, then a recipient is assured that any message that it receives comes from the alleged sender, because only that sender shares the relevant key. Such authentication is not inherent in a link encryption scheme. To achieve greater security, both link and end-to-end encryption are needed, as is shown in Figure 3.2.

When both forms of encryption are employed, the host encrypts the user data portion of a packet using an end-to-end encryption key. The entire packet is then encrypted using a link encryption key. As the packet traverses the network, each switch decrypts the packet, using a link encryption key to read the header, and then encrypts the entire packet again for sending it out on the next link. Now the entire packet is secure except for the time that the packet is actually in the memory of a packet switch, at which time the packet header is in the clear.

Logical Placement of End-to-End Encryption Function

With link encryption, the encryption function is performed at a low level of the communications hierarchy. In terms of the Open Systems Interconnection (OSI) model, link encryption occurs at either the physical or link layers. For end-to-end encryption, several choices are possible for the logical placement of the encryption function. At the lowest practical level, the encryption function could be performed at the network layer. Thus, for example, encryption could be associated with the frame relay or ATM protocol, so that the user data portion of all frames or ATM cells is encrypted.

With network-layer encryption, the number of identifiable and separately protected entities corresponds to the number of end systems in the network. Each end system can engage in an encrypted exchange with another end system if the two share a secret key. All the user processes and applications within each end system would employ the same encryption scheme with the same key to reach a particular target end system.

With this arrangement, it might be desirable to off-load the encryption function to some sort of front-end processor (typically a communications board in the end system). In this example, an electronic mail gateway is used to interconnect an internetwork that uses an OSI-based architecture with one that uses a TCP/IP-based architecture.

In such a configuration, there is no end-to-end protocol below the application layer. The transport and network connections from each end system terminate at the mail gateway, which sets up new transport and network connections to link to the other end system. Furthermore, such a scenario is not limited to the

case of a gateway between two different architectures. Even if both end systems use TCP/IP or OSI, there are plenty of instances in actual configurations in which mail gateways sit between otherwise isolated internetworks.

Thus, for applications like electronic mail that have a store-and forward capability, the only place to achieve end-to-end encryption is at the application layer.

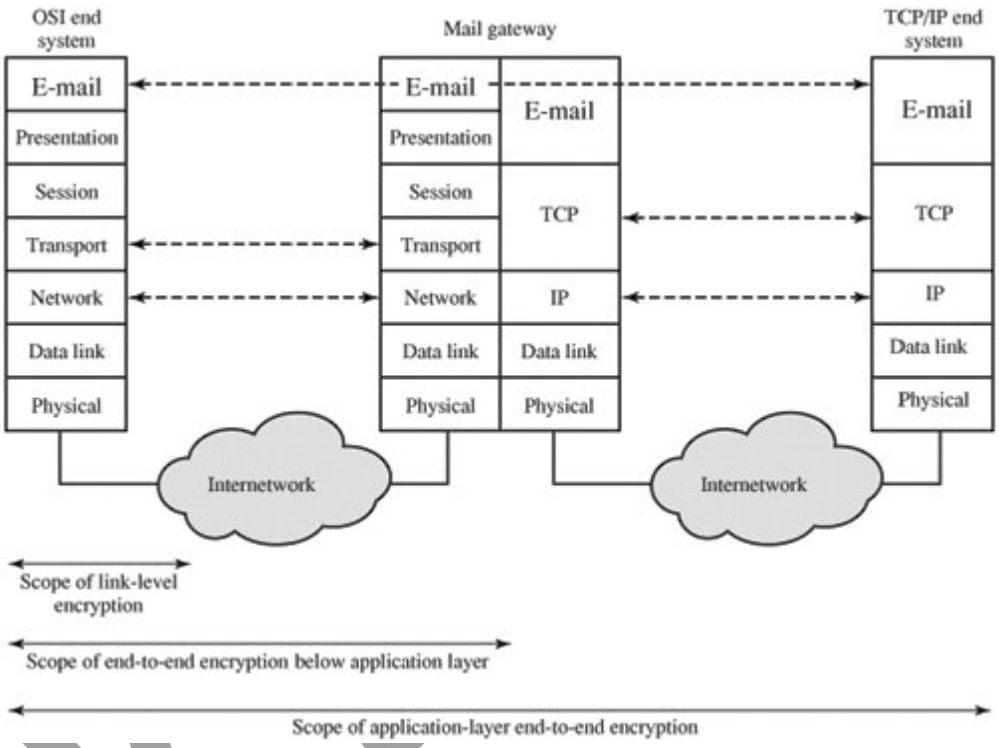


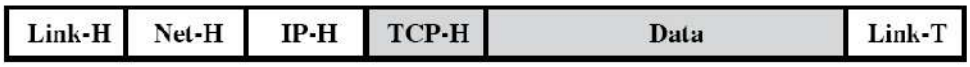
Fig 3.3. Encryption Coverage Implications of Store-and-Forward Communications

A drawback of application-layer encryption is that the number of entities to consider increases dramatically. A network that supports hundreds of hosts may support thousands of users and processes.

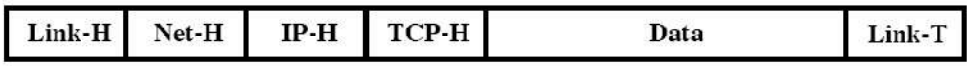
Thus, many more secret keys need to be generated and distributed.

An interesting way of viewing the alternatives is to note that as we move up the communications hierarchy, less information is encrypted but it is more secure. Figure 7.5 highlights this point, using the TCP/IP architecture as an example. In the figure, an application-level gateway refers to a store-and forward device that operates at the application level.

(a) Application-Level Encryption (on links and at routers and gateways)



On links and at routers



In gateways

(b) TCP-Level Encryption

On links

In routers and gateways

(c) Link-Level Encryption

Shading indicates encryption.

TCP-H = TCP header
IP-H = IP header
Net-H = Network-level header (e.g., X.25 packet header, LLC header)
Link H = Data link control protocol header
Link-T = Data link control protocol trailer

Fig 3.4 Encryption at different layers

With application-level encryption (Fig a), only the user data portion of a TCP segment is encrypted. The TCP, IP, network-level, and link-level headers and link-level trailer are in the clear. By contrast, if encryption is performed at the TCP level (Fig b), then, on a single end-to-end connection, the user data and the TCP header are encrypted.

The IP header remains in the clear because it is needed by routers to route the IP datagram from source to destination. Note, however, that if a message passes through a gateway, the TCP connection is terminated and a new transport connection is opened for the next hop.

Furthermore, the gateway is treated as a destination by the underlying IP. Thus, the encrypted portions of the data unit are decrypted at the gateway. If the next hop is over a TCP/IP network, then the user data and TCP header are encrypted again before transmission. However, in the gateway itself the data unit is buffered entirely in the clear. Finally, for link-level encryption (Figure 7.5c), the entire data unit except for the link header and trailer is encrypted on each link, but the entire data unit is in the clear at each router and gateway.

Traffic Confidentiality

We mentioned in Chapter 1 that, in some cases, users are concerned about security from traffic analysis. Knowledge about the number and length of messages between nodes may enable an opponent to determine who is talking to whom. This can have obvious implications in a military conflict. Even in commercial applications, traffic analysis may yield information that the traffic generators would like to conceal. [MUFT89] lists the following types of information that can be derived from a traffic analysis attack:

- Identities of partners
- How frequently the partners are communicating
- Message pattern, message length, or quantity of messages that suggest important information is being exchanged
- The events that correlate with special conversations between particular partners

Another concern related to traffic is the use of traffic patterns to create a **covert channel**. A covert channel is a means of communication in a fashion unintended by the designers of the communications facility. Typically, the channel is used to transfer information in a way that violates a security policy. For example, an employee may wish to communicate information to an outsider in a way that is not detected by management and that requires simple eavesdropping on the part of the outsider. The two participants could set up a code in which an apparently legitimate message of a less than a certain length represents binary zero, whereas a longer message represents a binary one. Other such schemes are possible.

Link Encryption Approach

With the use of link encryption, network-layer headers (e.g., frame or cell header) are encrypted, reducing the opportunity for traffic analysis. However, it is still possible in those circumstances for an attacker to assess the amount of traffic on a network and to observe the amount of traffic entering and leaving each end system.

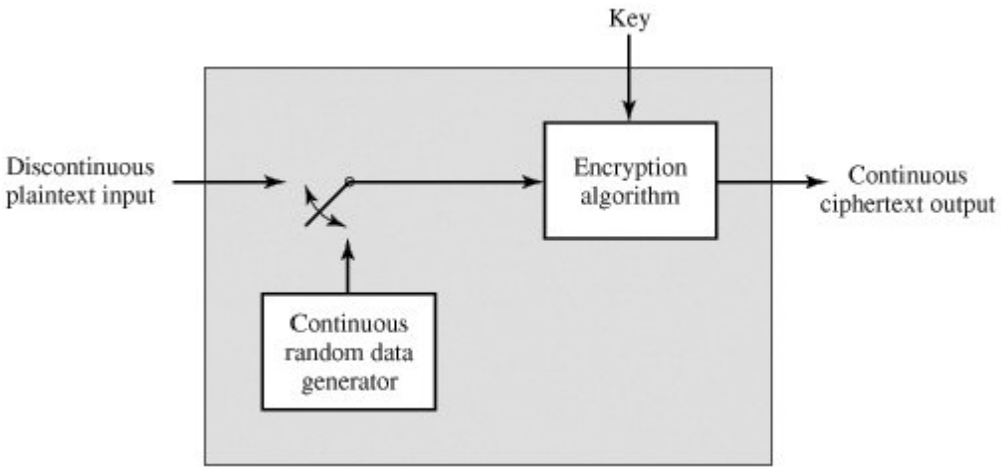


Fig 3.5 Traffic-Padding Encryption Device

Traffic padding produces ciphertext output continuously, even in the absence of plaintext. A continuous random data stream is generated. When plaintext is available, it is encrypted and transmitted. When input plaintext is not present, random data are encrypted and transmitted. This makes it impossible for an attacker to distinguish between true data flow and padding and therefore impossible to deduce the amount of traffic.

End-to-End Encryption Approach

Traffic padding is essentially a link encryption function. If only end-to-end encryption is employed, then the measures available to the defender are more limited. For example, if encryption is implemented at the application layer, then an opponent can determine which transport entities are engaged in dialogue. If encryption techniques are housed at the transport layer, then network-layer addresses and traffic patterns remain accessible. One technique that might prove useful is to pad out data units to a uniform length at either the transport or application level. In addition, null messages can be inserted randomly into the stream. These tactics deny an opponent knowledge about the amount of data exchanged between end users and obscure the underlying traffic pattern.

Key Distribution

• symmetric schemes require both parties to share a common secret key
• issue is how to securely distribute this key
• often secure system failure due to a break in the key distribution scheme

Key Distribution

• given parties A and B, there are various **key distribution** alternatives:
1. A can select key and physically deliver to B
2. third party can select & physically deliver key to A & B
3. if A & B have communicated previously, they can use previous key to encrypt a new key

4. if A & B have secure communications with a third party C, C can relay key between A & B (Key Distribution Center-KDC)

Key Distribution Issues

1. Hierarchical Key Control: hierarchies of KDC's required for large networks, but must trust each other-Local KDCs,
2. Session Key Lifetimes: should be limited for more security:
• new key for each session,
• Change the key periodically- if the session has long lifetime
3. Transparent Key Control Scheme: use of automatic key distribution on behalf of users, but must trust system.
4. Decentralized Key Control: use of decentralized key Distribution

Key Distribution

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. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key.

Therefore, the strength of any cryptographic system rests with the *key distribution technique*, a 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 a number 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 new key 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 on the encrypted links to A and B.

Options 1 and 2 call for manual delivery of a key. For link encryption, this is a reasonable requirement, because each link encryption device is going to be exchanging data only with its partner on the other end of the link. However, for end-to-end encryption, manual delivery is awkward. In a distributed system, any given host or terminal may need to engage in exchanges with many other hosts and terminals over time. Thus, each device needs a number of keys supplied dynamically. The problem is especially difficult in a wide area distributed system.

A Key Distribution Scenario

The key distribution concept can be deployed in a number of ways. The scenario assumes that each user shares a unique master key with the key distribution center (KDC).

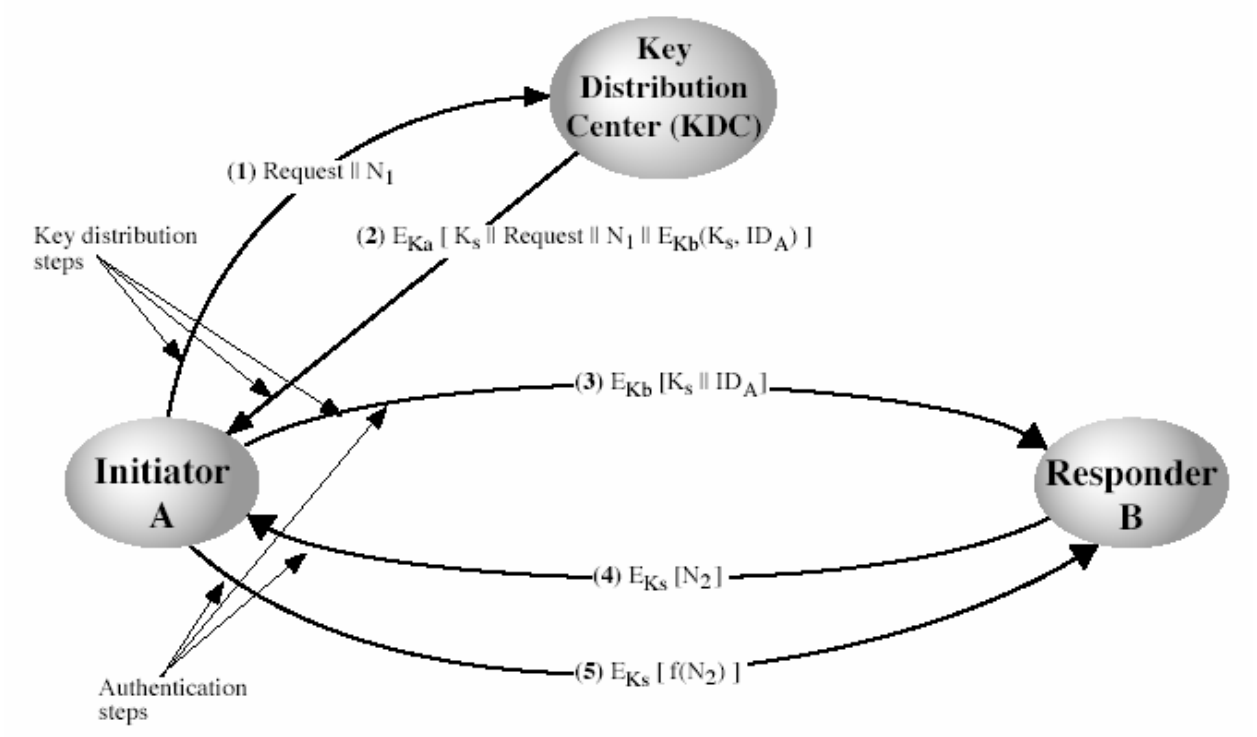


Fig 3.6 Key Distribution

Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection.

A has a master key, *Ka*, known only to itself and the KDC; similarly, B shares the master key *Kb* 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, *N1*, 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, K_s to be used for the session
- An identifier of A (e.g., its network address), ID_A

These last two items are encrypted with K_b (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 some transformation on N_2 (e.g., adding one).

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. For communication among entities within the same local domain, the local KDC is responsible for key distribution. If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC. In this case, any one of the three KDCs involved can actually select the key.

The hierarchical concept can be extended to three or even more layers, depending on the size 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. Furthermore, such a scheme limits the damage of a faulty or subverted KDC to its local area only.

Session Key Lifetime

The more frequently session keys are exchanged, the more secure they are, because the opponent has less ciphertext to work with for any given session key. On the other hand, 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 considerations in 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 new session. If a logical connection has a very long lifetime, then it would be prudent to change the session 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 no explicit 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. However, this negates one of the principal benefits of connectionless protocols, which is minimum overhead and delay for each transaction. A better strategy is to use a given session key for a certain fixed period only or for a certain number of transactions.

A Transparent Key Control Scheme

The scheme is 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 consists of functionality at one protocol layer, that performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

A transparent Key Control Scheme- Automatic Key Distribution

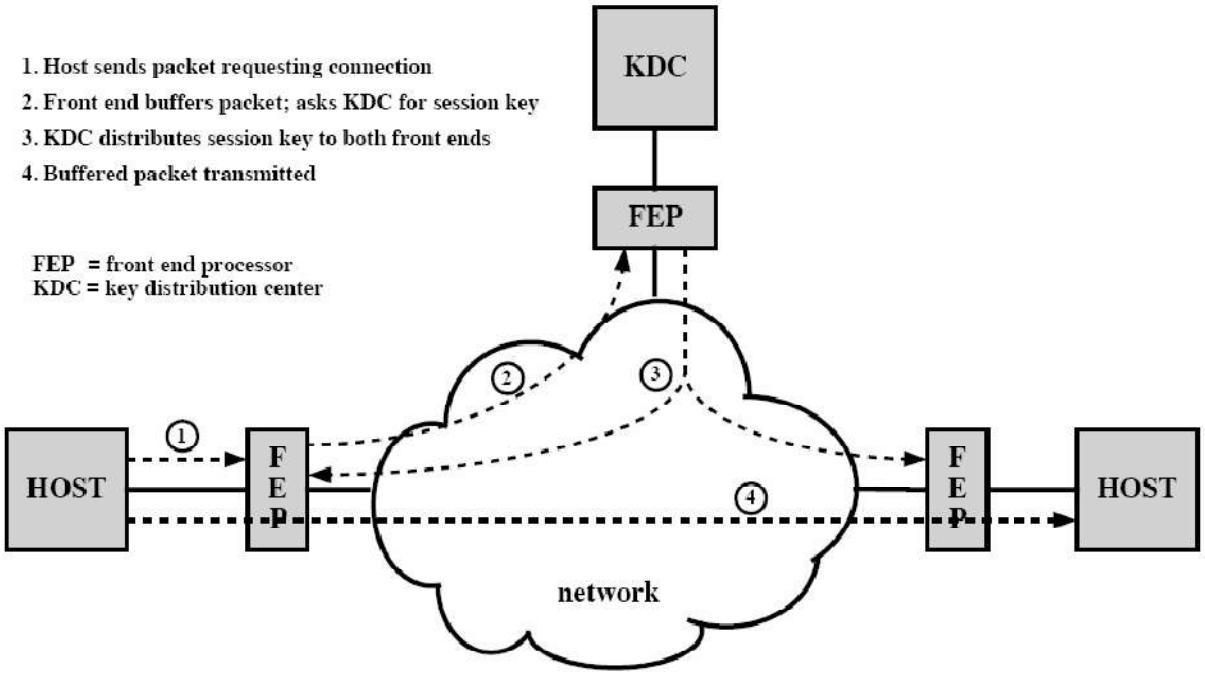


Fig 3.7. Automatic Key Distribution for Connection-Oriented Protocol

The steps involved in establishing a connection are shown in the figure. When one host wishes to set up a connection to another host, it transmits a connection-request packet (step 1).

The SSM saves that packet and applies to the KDC for permission to establish the connection (step 2).

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 (step 3).

The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems (step 4).

All user data exchanged between the two end systems are encrypted by their respective SSMs using the one-time session key. 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 distribution is 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 $[n(n\ 1)]/2$ master keys for a configuration with n end systems. A session key may be established with the following sequence of steps

1. A issues a request to B for a session key and includes a nonce, N_1

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), and another nonce, N_2.

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

4. Decentralized Key Control

• Might need to have n(n-1)/2 master keys. At most (n-1) stored at each node. Why?

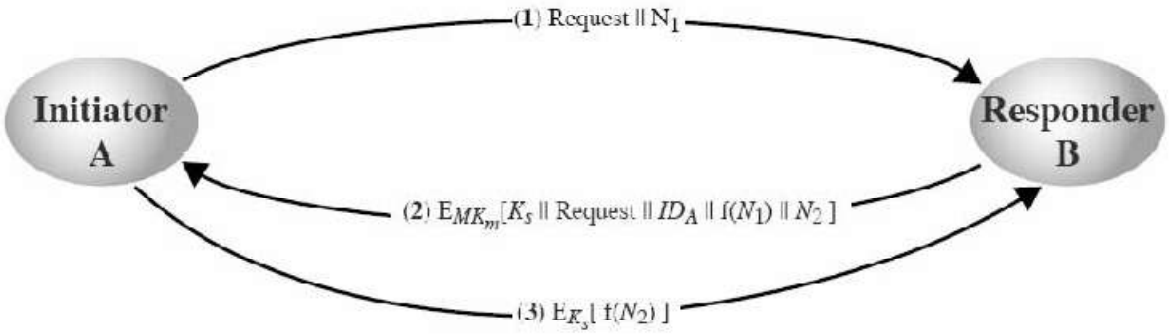


Fig 3.8 Decentralized Key Distribution

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 may also 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

The proposed technique is for use with DES and makes use of the extra 8 bits in each 64-bit DES key. That is, the 8 nonkey 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.

Public key Cryptography and RSA

Principles of Public-Key Cryptosystems

The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption.

As we have seen, key distribution under symmetric encryption requires either
(1) that two communicants already share a key, which somehow has been distributed to them; or
(2) the use of a key distribution center.

Whitfield Diffie, one of the discoverers of public-key encryption (along with Martin Hellman, both at Stanford University at the time), reasoned that this second requirement negated the very essence of cryptography: the ability to maintain total secrecy over your own communication.

Diffie and Hellman achieved an astounding breakthrough in 1976 by coming up with a method that addressed both problems and that was radically different from all previous approaches to cryptography, going back over four millennia

Public-Key Cryptosystems

Asymmetric algorithms rely on one key for encryption and a different but related key for decryption.

These algorithms have the following important characteristic:

- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

In addition, some algorithms, such as RSA, also exhibit the following characteristic:

- Either of the two related keys can be used for encryption, with the other used for decryption.

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.

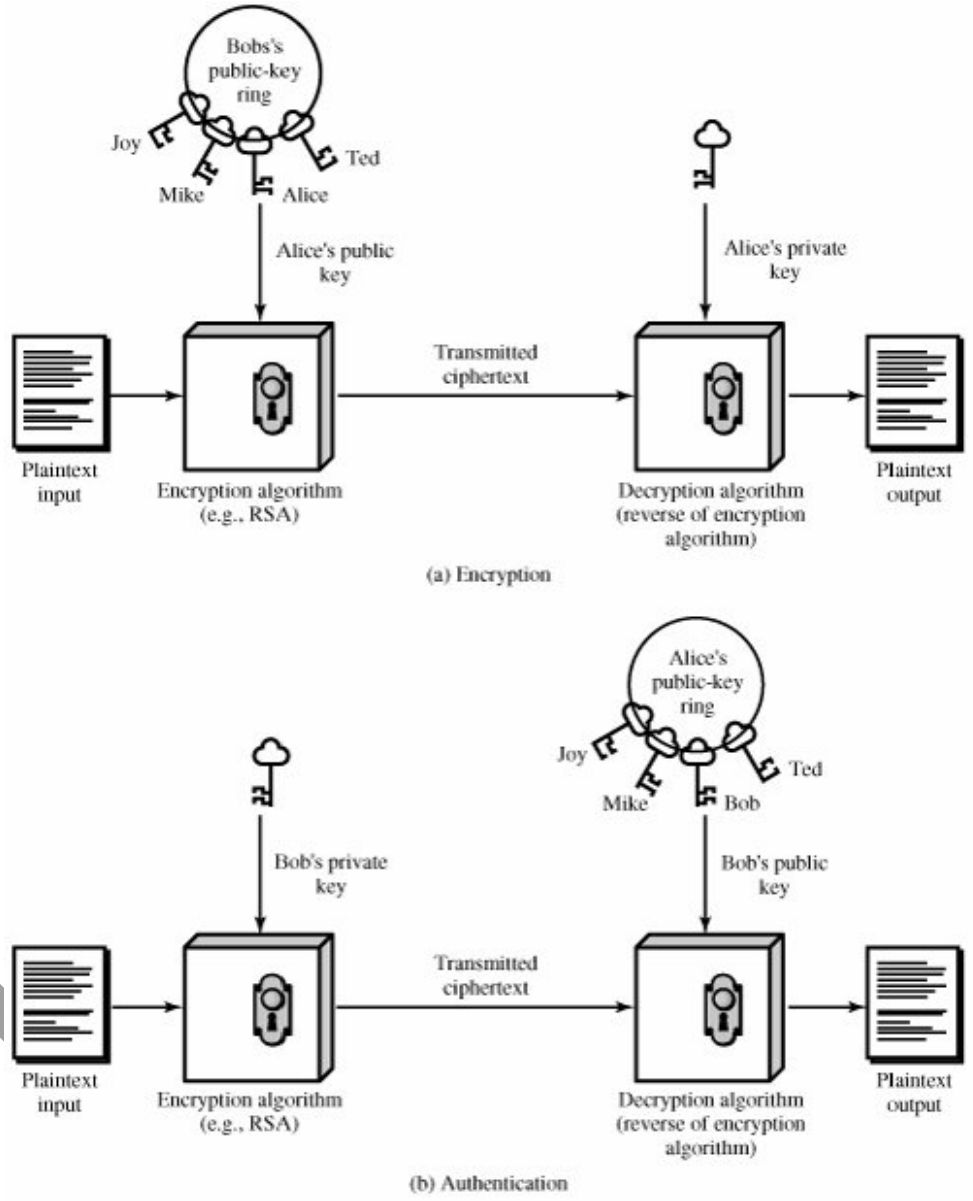


Fig 3.9 Public key cryptography

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. As Figure 9.1a suggests, 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.

With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user's private key remains protected and secret, incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

Applications for Public-Key Cryptosystems

Before proceeding, we need to clarify one aspect of public-key cryptosystems that is otherwise likely to lead to confusion. Public-key systems are characterized by the use of a cryptographic algorithm with two keys, one held private and one available publicly. Depending on the application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function. In broad terms, we can classify the use of public-key cryptosystems into three categories:

- **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 or to a small block of data that is a function of the message.
- **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

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 OF PUBLIC KEY CRYPTOGRAPHY

The cryptosystem illustrated in Figures 10.3 through 10.5 depends on a cryptographic algorithm based on two related keys. Diffie and Hellman postulated this system without demonstrating that such algorithms exist. However, they did lay out the conditions that such algorithms must fulfill.

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 cipher text: $C = E(PUb, M)$

3. It is computationally easy for the receiver B to decrypt the resulting cipher text 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 opponent, knowing the public key, *PUb*, to determine the private key, *PRb*.

5. It is computationally infeasible for an opponent, knowing the public key, *PUb*, and a cipher text, *C*, to recover the original message, *M*.

We can add a sixth requirement that, although useful, is not necessary for all public-key applications.

6. The two keys can be applied in either order: $M = D[PUb, E(PRb,M)] = D[PRb, E(PUb, M)]$

These are formidable requirements, as evidenced by the fact that only a few algorithms (RSA, elliptic curve cryptography, Diffie-Hellman, DSS) have received widespread acceptance in the several decades since the concept of public-key cryptography was proposed.

Security of Public Key Schemes

- like private key schemes brute force **exhaustive search** attack is always theoretically possible
- but keys used are too large (>512bits)
- security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems
- more generally the **hard** problem is known, but is made hard enough to be impractical to break
- requires the use of **very large numbers**
- hence is **slow** compared to private key schemes

The RSA Algorithm

Diffie and Hellamn [1976] introduced an approach for public key systems and challenged researchers to devise algorithms that met the requirements. The challenge was met by algorithm

proposed by Ron Rivest, Adi Shamir, Len Adleman (RSA) in 1978 at MIT.

RSA is block cipher in which plain text and cipher text are integers between 0 and n-1 for some n. Typical n has 1024 bits ($n < 2^{1024}$) or 309 decimal digits.

Details of the algorithm

RSA makes use of an expression with exponentials. 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 $\log_2 (n + 1)$. In practice, the block size is i bits, where $2^i < n < 2^{i+1}$.

Encryption and decryption are of the following form, for some plaintext block M and cipher text block C.

$$C = M^e \bmod n$$
$$M = C^d \bmod n$$
$$= M^{ed} \bmod n$$

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d.
Thus, this is a public-key encryption algorithm with a public key $PU = \{e, n\}$ and a private key $PR = \{d, n\}$.

For this algorithm to be satisfactory for public-key encryption, the following requirements must be met.

1. It is possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M < n$.
2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$.
3. It is infeasible to determine d given e and n.

For now, we focus on the first requirement and consider the other questions later. We need to find a relationship of the form $M^{ed} \bmod n = M$.
The preceding relationship holds if e and d are multiplicative inverses modulo $\varphi(n)$, where $\varphi(n)$ is the Euler totient function. It is shown that for p, q prime, $\varphi(pq) = (p - 1)(q - 1)$.
The relationship between e and d can be expressed as

$$e^d \bmod \varphi(n) = 1 \text{ ----------(11.1)}$$

This is equivalent to saying

$$e^d \equiv 1 \bmod \Box(n)$$
$$d \equiv e^{-1} \bmod \Box(n)$$

That is, *e* and *d* are multiplicative inverses mod □ (n). Note that, according to the rules of modular
arithmetic, this is true only if *d* (and therefore *e*) is relatively prime to □ (n). Equivalently, gcd(□ (n), *d*) = 1. The equation (11.1) satisfies the requirement for RSA.

We are now ready to state the RSA scheme. The ingredients are the following:
1. p, q be two prime numbers (private, chosen)
2. n=p*q (public, calculated)
3. Select e, with gcd(□ (n),e)=1; 1 < e < □ (n) (public, chosen)
4. Calculate d=e^{-1} mod □ (n) (private, calculated)

The private key consists of {*d, n*} and the public key consists of {*e, n*}. Suppose that user A has published his public key and that user B wishes to send the message *M* to A.
Then B calculates $C = M^e \bmod n$ and transmits *C*.
On receipt of this ciphertext, user A decrypts by calculating
$M = C^d \bmod n$.

Suppose A wants to send an encrypted message to B, then the sequence of exchanges between the two are given as follows:

RSA Algorithm - Method 1

1. Key generation by B

a. Select p and q that are prime numbers and p≠q.
b. Calculate n = p * q.
c. Calculate □ (n) = (p-1)(q-1).
d. Select integer e that is prime to □ (n) and less than □ (n).
e. Calculate $d = e^{-1} \pmod{\Box(n)}$.
f. $PU_B = \{e, n\}$.
g. $PR_B = \{d, n\}$.

2. Encryption by A with B's public key

a. Plain text be M < n.
b. Cipher text is $C = M^e \bmod n$

3. Decryption by B with B's private key

a. Cipher text is C.

b. Retrieved plain text is C^d mod n.

Method 2

1. Key generation by A

a. Select p, q that are prime numbers and p≠q.
b. Calculate n = p * q.
c. Calculate □ (n) = (p-1)(q-1).
d. Select integer e that is prime to □ (n) and less than □ (n).
e. Calculate d = e^{-1} (mod □ (n)).
f. PU_A = {e, n}.
g. PR_A= {d, n}.

2. Encryption by A with A's private key

a. Plaintext be M < n.
b. Cipher text is C = M^d mod n

3. Decryption by B with A's public key

a. Cipher text is C
b. Retrieved plaintext is C^e mod n.

EXAMPLES OF ENCRYPTION/ DECRYPTION

I. Working example

1. Select p=17, q=11
2. Calculate n=p*q=17 x 11= 187
3. Calculate □ (n) =(p-1)*(q-1)=16 x 10 = 160
4. Select e such that e is relatively prime to 160 and < 160 say e=7
5. Determine d such that d*e □ 1 mod 160 and d<160 (value of d =23 (since 23 x7= 161)

The keys are PU= [7, 187] and PR= [23, 187].

Let plain text be M=88. For encryption we need to calculate 88^7 mod 187 =11 (cipher) and for decryption we need to calculate 11^{23} mod 187 = 88 (plain text). Thus plain text is successfully recovered during decryption.

II. Example where plain text not recovered

1. Suppose n=187 (7 bit number), e=7 and d(calculated inverse) =23
2. Let M=189 (7 bits)
3. We get C=189_7 mod 187 = 128 (encryption)
4. M=128^{23} mod 187 = 2 (decryption)

Note that retrieved M (2) is different from actual M (189). The problem is because M is not < n. In this example, n is a 7 bit number. Any 7 bit number cannot be M. In other words, block size cannot be taken as the number of bits in n.

Encryption/Decryption computation

Both encryption and decryption in RSA involve raising an integer to an integer power, mod *n*. If the exponentiation is done over the integers and then reduced modulo *n*, the intermediate values would be very huge. Fortunately, as the preceding example shows, we can make use of a property of modular arithmetic given here to reduce complexity.

$(a \times b) \bmod n = [(a \bmod n) \times (b \bmod n)] \bmod n$

Thus, we can reduce intermediate results modulo *n*. This makes the calculation practical.

SECURITY OF RSA

Four possible approaches to attacking the RSA algorithm are

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

PRIME NUMBERS

For many cryptographic algorithms we need large prime numbers. So, we generate a random number and then test if it is prime. Deterministic algorithm for prime number test is complex (check if any number in the range of 2 to √n divides n; if so it is not prime). Miller (1975) and Rabin (1980) developed efficient algorithm that almost certainly determines if n is

prime. More results on number theory are needed to understand Miller Rabin method. Before discussing the algorithm that tests a given number to be prime, we state some results on prime numbers supported by examples.

Results on prime numbers

1. Any odd positive number n >2 can be expressed as $n-1=2^k q$ where k>0 and q is odd

Example 1: Let n=23 (prime), n-1 = 22 = 2 x 11 (k=1 and q=11)

122

Example 2: Let n=35 9non prime), n-1 = 34 = 2 x 17 (k=1 and q=17)

2. Property 1 of prime number: If p is prime and a is any positive integer then $a^2 \bmod p = 1$ iff either a mod p = 1 or a mod p = -1(p-1)

Example 3: Let a=4, p=3; $a^2 \bmod p = 16 \bmod 3 = 1$;

Also a mod p =1

Example 4: Let a=4, p=2; $a^2 \bmod p = 16 \bmod 2 = 0$;

Neither 4 mod 2 = 1 nor 4 mod 2 = -1

Example 5: Let a=6, p=7; a_2 mod p = 36 mod 7 = 1

Also a mod p = -1

3. Property 2 of prime number: Let p be prime > 2. Recall p-1 = 2^k q where k >0, q odd. Let a be any integer such that 1 < a < p-1 then one of the following conditions is true.

i. a_q mod p =1

ii. One of a_q, a_{2q}, … $a_{(2^{(k-1)*q})}$ is congruent to -1 mod p. That is there is some number 1 < j < k so that a_j = -1 mod p.

Example 6: Suppose p=5, a=3; p-1= 4 and k = 2, q=1. a_q mod p = 3 mod 5 is not 1

But a_{2q} mod p = 9 mod 5 = 4 = -1 mod 5

Example 7: Suppose p= 7, a=4; p-1=6 and k=1, q=3. a_q mod p = 64 mod 7 =1

DISCRETE LOGARITHMS

The Discrete logarithm is fundamental to many public key encoding systems including Diffel Hellman key exchange algorithms and digital signatures.

Now consider more general expression a_m = 1 mod n. If a and n are relatively prime there is at least one integer (□ (n)) that satisfies the general expression.

The least positive integer m which satisfies the equation a_m = 1 mod n is called (i) order of a mod n (ii) exponent to which *a* belongs to mod n (iii) length of the period generated by *a*

Example 8:

Consider a=7, n= 19

7^1 = 7mod 19

7^2 =11 mod 19

7^3 = 1 mod 19

7^4 = 7 mod 19 …

The sequence repeats. The period is 3. This is nothing but the smallest integer m for which a_m = 1 mod n (in this example 7_3 = 1 mod 19).

For a = 2, you can check that (a, a_2, a_3, … $a_{(19)}$) mod 19 are 2, 4, 8, 16, 13, 7, 14, 9, 18, 17, 15, 11, 3, 6, 12, 5, 10, 1

All numbers 1 to 18 (the value of (19)) have appeared. The period is 18, full period. Similarly you can verify that 3, 10, 13, 14, 15 all have full period.

Primitive root

In general the highest possible exponent to whom a number can belong (mod n) is □ (n). If a number is of this order it is referred to as a primitive root of n. Alternatively if a is a primitive root of n then a, a_2, a_3, … $a_{(n)}$ are all distinct (mod n) and are all prime to n. In particular for a prime number *p*, if *a* is a primitive root of *p* then a, a_2, a_3, … a_{p-1} are distinct (mod p) and prime

to p. from the preceding example we see that, primitive roots of 19 are 2, 3, 10, 13, 14 and 15. Table 9.1 shows all the powers of a, modulo 19 for all positive a < 19. The length of the sequence for each base value is indicated by shading. Note the following:

1. All sequences end in 1. This is consistent with the reasoning of the preceding few paragraphs.

2. The length of a sequence divides □(19) =18. That is, an integral number of sequences occur in each row of the table. The length of the sequence for a = 1 is 1. For a = 2 and 3 it is 18. For a = 4 it is 9 and so on. Note that all these lengths divide □ (n) (=18).

3. Some of the sequences are of length 18. In this case, it is said that the base integer a generates (via powers) the set of nonzero integers modulo 19. Each such integer is called a primitive root of the modulus 19. Some primitive roots of 19 are 2, 3, 10.

a	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^{10}	a^{11}	a^{12}	a^{13}	a^{14}	a^{15}	a^{16}	a^{17}	a^{18}
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2	4	8	16	13	17	14	9	18	17	15	11	3	6	12	5	10	1
3	9	8	5	15	7	2	6	18	16	10	11	14	4	12	17	13	1
4	16	7	9	7	11	6	5	1	4	16	7	9	17	11	6	5	1
5	6	11	17	9	7	6	4	1	5	6	11	17	9	7	11	4	1
6	17	7	4	5	11	9	16	1	6	17	7	4	5	11	9	16	1
7	11	1	7	11	1	7	11	1	7	11	1	7	11	1	7	11	1
8	7	18	11	12	1	8	7	18	11	12	1	8	7	18	11	12	1
9	5	7	6	16	11	4	17	1	9	5	7	6	16	11	4	17	1
10	5	12	6	3	11	15	17	18	19	14	7	13	16	8	4	2	1
11	7	1	11	7	1	11	7	1	11	7	1	11	7	1	11	7	1
12	11	18	7	8	1	12	11	18	7	8	1	12	11	18	7	8	1
13	17	12	4	14	11	10	16	18	6	2	7	15	5	8	9	3	1
14	6	8	17	10	7	3	4	18	5	13	11	2	9	12	16	15	1
15	16	12	9	2	11	13	5	18	4	3	7	10	17	8	6	14	1
16	9	11	5	4	7	17	6	1	16	9	11	5	4	7	17	6	1
17	4	11	16	6	7	5	9	1	17	4	11	16	6	7	5	9	1
18	1	18	1	18	15	18	1	18	1	18	1	18	1	18	1	18	1

Powers of integers modulo 19

More generally, we can say that the highest possible exponent to whom a number can belong (mod n) is □ (n). If a number is of this order, it is referred to as a primitive root of n. The importance of this notion is that if a is a primitive root of n, then its powers a, a_2, a_3, … $a^{□(n)}$ are distinct (mod n) and are all relatively prime to n. In particular, for a prime number p, if a is a primitive root of p, then a, a_2, a_3, … a_{p-1} are distinct (mod p). For the prime number 19, its primitive roots are 2, 3, 10, 13, 14, and 15.

Not all integers have primitive roots. In fact, the only integers with primitive roots are those of the form 2, 4, p^{α} and $p_{2\alpha}$, where p is any odd prime and α is a positive integer.

Logarithms for modular arithmetic

We review some properties of ordinary logarithms here.
$\log_x(1) = 0$, $\log_x(x) = 1$
$\log_x(yz) = \log_x(y) * \log_x(z)$
$\log_x(y/z) = \log_x(y) - \log_x(z)$
$\log_x(r_a) = a * \log_x(r)$
Consider the primitive root a for some prime number p. We know that powers of a from 1 to *p*-1 are distinct and are integers 1 to p-1 in some order. We also know that any integer b satisfies b□r mod p for some r, where $0 \le r \le p-1$ by definition of modular arithmetic.
Let *a* be a primitive root of *p*. For the integer b we can find a unique exponent i such that $b \equiv a^i \bmod p$, where $0 \le i \le p-1$. The same equation is written in terms of logarithm as $\text{dlog}_{a,p}(b) = i$.
The exponent i is called the discrete logarithm of the number b for the base a mod p.
All the properties of ordinary logarithms are satisfied by discrete logarithms.
Example 9: Let p=19. We know that 2 is a primitive root of 19.
18 1 18 1 18 15 18 1 18 1 18 1 18 1 18 1 18 1
Consider b = 52. 52 mod 19 = 14 = 2^7 mod 19
Thus $\text{dlog}_{2,19}(52) = 7$
For 19, 3 is another primitive root
Consider b=52 and 52 mod 19 = 14 = 3^{13} mod 19
Thus $\text{dlog}_{3,19}(52) = 13$
Discrete logarithms can also be defined for non prime bases. All we need is a primitive root.
Consider n=9. □ (9) = 6 and a =2 is a primitive root since (2^1, 2^2, … 2^6)mod 9 are distinct and prime to 9. These are nothing but (2, 4, 8, 7, 5, 1)
The logarithm table is given here

No.	2	4	8	7	5	1
$\text{Dlog}_{2,9}$	1	2	3	4	5	6/0

Unlike prime number we don□t have logarithm for all numbers. For example, discrete logarithm for number 3, 6 are not defined.
Note that the properties of ordinary logarithms are true in case of discrete logarithms too. We now show some of the properties of discrete logarithms.
$\text{dlog}_{a,p}(1) = 0$ because $a^0 \bmod p = 1 \bmod p = 1$
$\text{dlog}_{a,p}(a) = 1$ because $a^1 \bmod p = a$

Calculation of discrete logarithms
Consider the equation $y = g^x \bmod p$. Given g, x and p it is straight forward to calculate y.
We can multiply *g* with itself *x* times or find modulo at in between steps and find *y*. However given *y, g, p* it is difficult to find *x* (dlog). This is as difficult as factoring a large number into product of primes.

DIFFIE HELLMAN KEY EXCHANGE

- first public-key type scheme proposed
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
  - note: now know that Williamson (UK CESG) secretly proposed the concept in 1970
- is a practical method for public exchange of a secret key
- used in a number of commercial products
- a public-key distribution scheme
  - cannot be used to exchange an arbitrary message
  - rather it can establish a common key
  - known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial) - easy
- security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

This is the first published public key algorithm [1976]. A lot of commercial products use this algorithm for key exchange.

Algorithm:

Global public: q, a prime number

a, a primitive root of q

User A key generation: Select private X_A with $X_A < q$

Calculate public Y_A as $Y_A = a^{X_A} \bmod q$

User B key generation: Select private X_B with $X_B < q$

Calculate public Y_B as $Y_B = a^{XB} \bmod q$

Calculation of secret key by user A: $K = Y_B^{XA} \bmod q$

Calculation of secret key by user B: $K = Y_A^{XB} \bmod q$

Secret Key Calculation:

Both A and B calculate secret key. The calculations of both give the same value.

$K = (Y_B)^{XA} \bmod q$ (key of A)

$= (a^{XB} \bmod q)^{XA} \bmod q$

$= (a^{XB})^{XA} \bmod q$

$= a^{XB * XA} \bmod q$

$= (a^{XA})^{XB} \bmod q$

$= (a^{XA} \bmod q)^{XB} \bmod q$

$= (Y_A)^{XB} \bmod q$ (key of B)

Security of Diffie Hellman key exchange

It is easy to calculate exponentials modulo a prime number and difficult to calculate discrete logarithms. For large prime numbers it is close to infeasible.

Example 1:

Let q=353, a=3 (primitive root of 353)

Suppose that A and B select secret keys X_A =97 and X_B =233.

Both compute public keys $Y_A = 3^{97} \bmod 353 = 40$ and $Y_B = 3^{233} \bmod 353 = 248$

They exchange public keys

Common secret key computed by A and B as

$Y_B^{XA} \bmod q = 248^{97} \bmod 353 = 160$ and

$(Y_A)^{XB} \bmod q = 40^{233} \bmod 353 = 160$

Attacker has access to q= 353, Y_A =40 and Y_B = 248

In this simple example, the attacker can find secret key 160 by brute force.

Attacker can find secret key by solving $3^a \bmod 353 = 40$ or $3^b \bmod 353 = 248$. a=97 is found (by systematic testing of $3^a \bmod 353 = 40$) which is the private key of A.

Now secret key is computed using $(Y_B)^{XA} \bmod q = 248^{97} \bmod 353 = 160$

Note that prime number is small in this example
In reality for large numbers finding a ($d_{3,353}$ log (40)) is difficult

Key exchange protocols

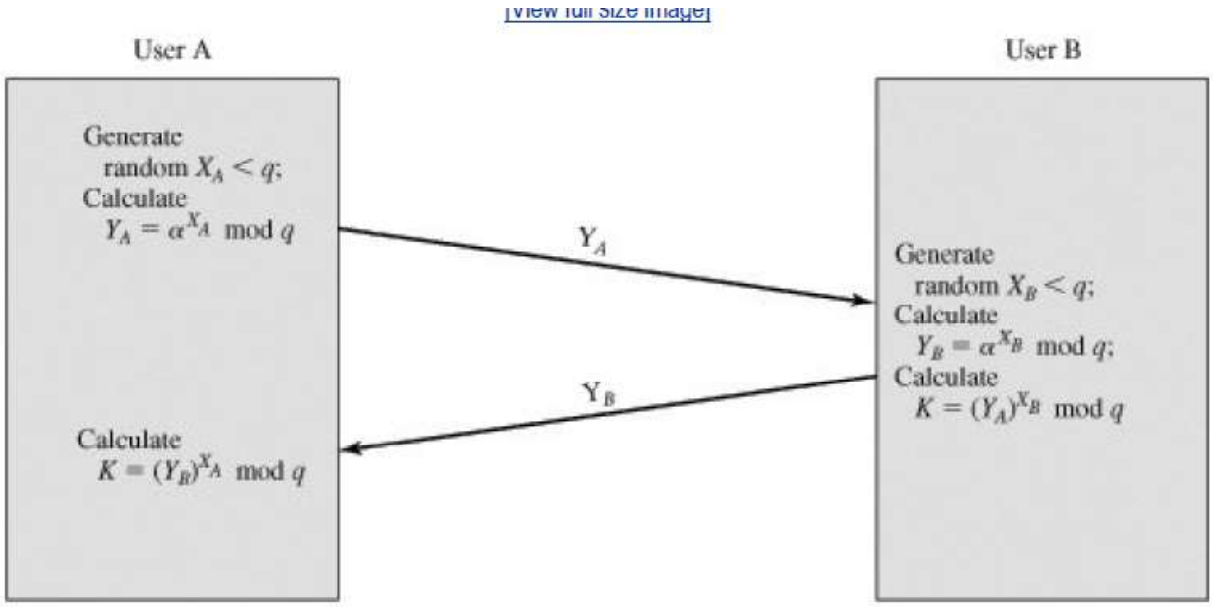
Suppose that A wishes to set up a connection with B and use a secret key to encrypt messages on that connection. The steps followed are

1. A and B know q, a
2. User A:
    2.1. Generates random $X_A < q$. Calculates $Y_A = a^{XA} \bmod q$
    2.2. Sends Y_A to B
3. User B

    1.1 Generates random $X_B < q$. Calculates $Y_B = a^{XB} \bmod q$ and the secret key $K= (Y_A)^{XB} \bmod q$
    1.2 Sends Y_B to A
2. A calculates $K= (Y_B)^{XA} \bmod q$

Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection. User A can generate a one-time private key X_A, calculate Y_A, and send that to user B. User B responds by generating a private value X_B calculating Y_B, and sending Y_B to user A. Both users can now calculate the key. The necessary public values q would need to be known ahead of time.

Alternatively, user A could pick values for q include those in the first message.



Attacks

A specific attack called Man in the middle attack is possible. Suppose A and B wish to exchange keys, and D is the adversary. The attack proceeds as follows.

1. D prepares for the attack by generating two random numbers for private keys say X_{D1}, X_{D2} and then computes public keys Y_{D1}, Y_{D2}
2. A transmits Y_A to B
3. D intercepts Y_A and sends Y_{D1} to B and calculates $K2 = (Y_A)^{X_{D2}} \bmod q$
4. B receives Y_{D1} and calculates $K1 = (Y_{D1})^{X_B} \bmod q$
5. B transmits Y_B to A
6. D intercepts Y_B and transmits Y_{D2} to A. D also calculates $K1 = (Y_B)^{X_{D1}} \bmod q$ and $K2 = (Y_A)^{X_{D2}} \bmod q$
7. A receives Y_{D2} and calculates $K2 = (Y_{D2})^{X_A} \bmod q$

At this point A and B think that they share a secret key, but B and D share the key K1 and A and D share the key K2. All future communications between A and B are compromised in the following way

1. A sends encrypted message M using K2 to B as E(K2, M).
2. D intercepts the encrypted message and recovers M
3. D sends E(K1, M) or E(K1, M□) to B, where M□ is any message.

Such attacks are possible because it does not authenticate the participants. Such attacks can be overcome with digital signatures.

Key Management

One of the major roles of public-key encryption has been to address the problem of key distribution. There are actually two distinct aspects to the use of public-key cryptography in this regard:

- The distribution of public keys
- The use of public-key encryption to distribute secret keys

We examine each of these areas in turn.

Distribution of Public Keys

Several techniques have been proposed for the distribution of public keys. Virtually all these proposals can be grouped into the following general schemes:

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

Public Announcement of Public Keys

On the face of it, the point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large (Figure 10.1). For example, because of the growing popularity of PGP (pretty good privacy, discussed in Chapter 15), which makes use of RSA, many PGP users have adopted the practice of appending their public key to messages that they send to public forums, such as USENET newsgroups and Internet mailing lists.

Figure 3.10. Uncontrolled Public-Key Distribution

Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication (see Figure 9.3).

Publicly Available Directory

A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys. Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization (Figure 10.2). Such a scheme would include the following elements:

1. The authority maintains a directory with a {name, public key} entry for each participant.

2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.

3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.

4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

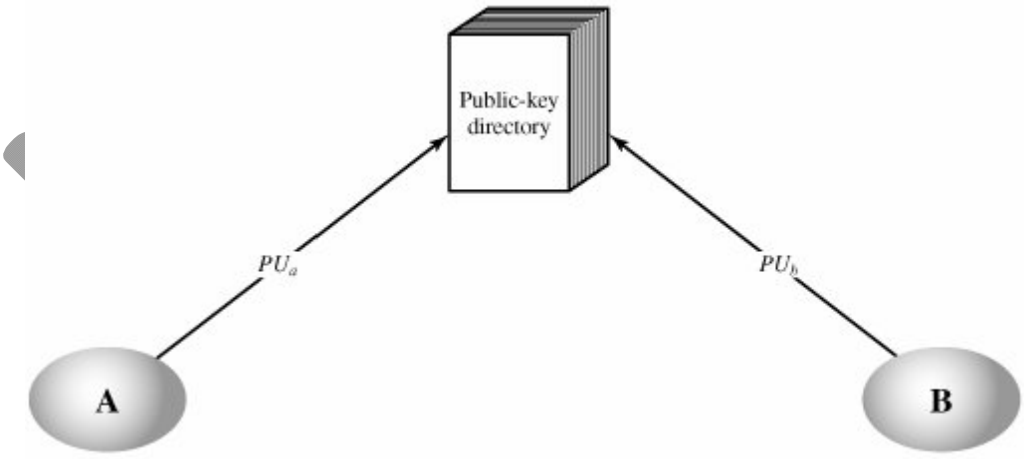


Figure 3.11. Public-Key Publication

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. A typical scenario is illustrated in Figure 10.3, which is based on a figure in [POPE79]. As before, the scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, each participant reliably knows a public

key for the authority, with only the authority knowing the corresponding private key. The following steps (matched by number to Figure 10.3) occur:

1. A sends a time stamped 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, *PRauth* 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, *PUb* 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 (*IDA*) 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 *PUa* 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.

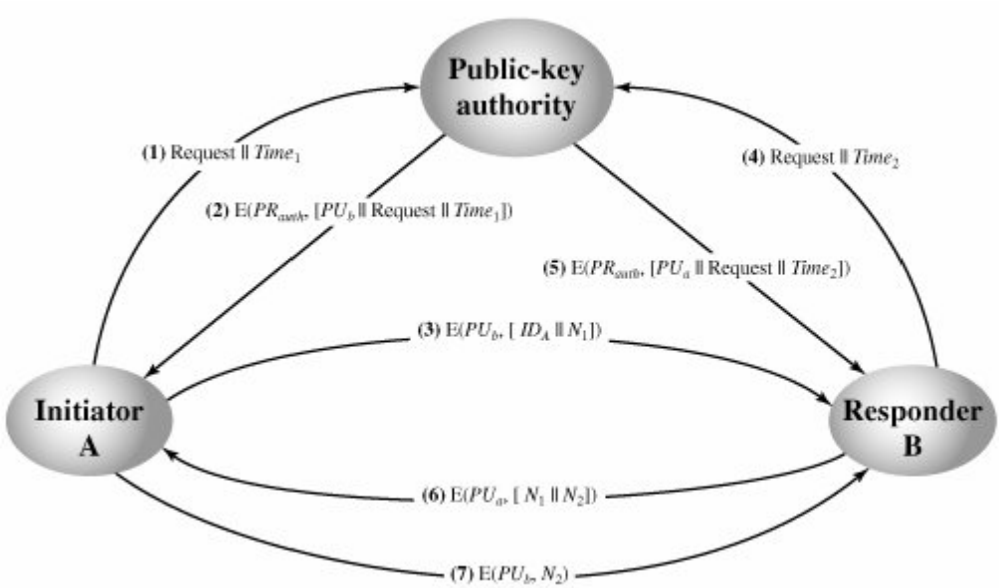


Figure 3.12. Public-Key Distribution Scenario

Thus, a total of seven messages are required. However, the initial four messages need be used only infrequently because both A and B can save the other's public key for future use, a technique known as caching. Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

Public-Key Certificates

The scenario of Figure 10.3 is attractive, yet it has some drawbacks. The public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact. As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

An alternative approach, first suggested by Kohnfelder [KOHN78], is to use **certificates** that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. In essence, a certificate consists of a public key plus an identifier of the key owner, with the whole block signed by a trusted third party.

Typically, the third party is a certificate authority, such as a government agency or a financial institution, that is trusted by the user community. A user can present his or her public key to the authority in a secure manner, and obtain a certificate. The user can then publish the certificate. Anyone needed this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. 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. These requirements are satisfied by the original proposal in [KOHN78]. Denning [DENN83] added the following additional requirement:
4. Any participant can verify the currency of the certificate.

A certificate scheme is illustrated in Figure 10.4. Each participant applies to the certificate authority, supplying a public key and requesting a certificate.

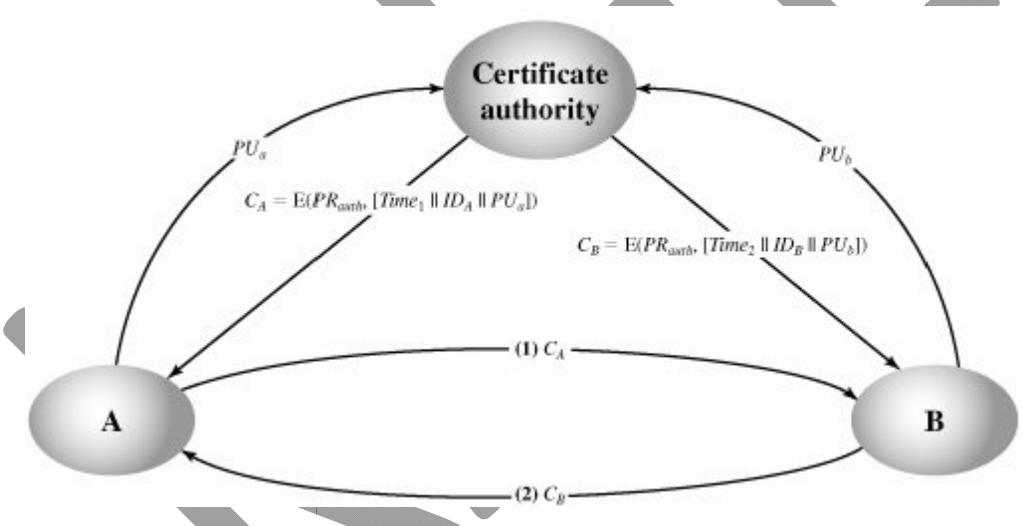


Figure 3.13. Exchange of Public-Key Certificates

Application must be in person or by some form of secure authenticated communication. For participant A, the authority provides a certificate of the form

$CA = E(PRauth, [T||IDA||PUa])$

where *PRauth* is the private key used by the authority and *T* is a timestamp. A may then pass this certificate on to any other participant, who reads and verifies the certificate as follows:

$D(PUauth, CA) = D(PUauth, E(PRauth, [T||IDA||PUa])) = (T||IDA||PUa)$

The recipient uses the authority's public key, *PUauth* to decrypt the certificate. Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority. The elements *IDA* and *PUa* provide the recipient with the name and public key of the certificate's holder. The timestamp *T* validates the currency of the certificate. The timestamp counters the following scenario. A's private key is learned by an adversary. A generates a new private/public key pair and applies to the certificate authority for a new certificate.

Meanwhile, the adversary replays the old certificate to B. If B then encrypts messages using the compromised old public key, the adversary can read those messages. In this context, the compromise of a private key is comparable to the loss of a credit card. The owner cancels the credit card number but is at risk until all possible communicants are aware that the old credit card is obsolete. Thus, the timestamp serves as something like an expiration date. If a certificate is sufficiently old, it is assumed to be expired.

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, all of which are discussed in Part Two. X.509 is examined in detail in Chapter 14.

Distribution of Secret Keys Using Public-Key Cryptography

Once public keys have been distributed or have become accessible, secure communication that thwarts eavesdropping (Figure 9.2), tampering (Figure 9.3), or both (Figure 9.4) is possible. However, few users will wish to make exclusive use of public-key encryption for communication because of the relatively slow data rates that can be achieved. Accordingly, public-key encryption provides for the distribution of secret keys to be used for conventional encryption.

Simple Secret Key Distribution

An extremely simple scheme was put forward by Merkle [MERK79], as illustrated in Figure 10.5. If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair {*PUa*, *PR*a} and transmits a message to B consisting of *PUa*
and an identifier of *A*, *IDA*.
2. B generates a secret key, *Ks*, and transmits it to A, encrypted with A's public key.
3. 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*.
4. A discards *PUa* and *PRa* and B discards *PUa*.

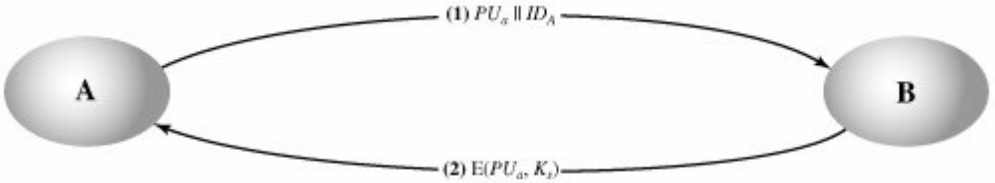


Figure 3.14. Simple Use of Public-Key Encryption to Establish a Session Key

A and B can now securely communicate using conventional encryption and the session key *Ks*. At the completion of the exchange, both A and B discard *Ks*. Despite its simplicity, this is an attractive protocol. No keys exist before the start of the communication and none exist after the completion of communication. Thus, the risk of compromise of the keys is minimal. At the same time, the communication is secure from eavesdropping. The protocol depicted in Figure 10.5 is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message (see Figure 1.4c). Such an attack is known as a **man-in-the-middle attack** [RIVE84]. In this case, if an adversary, E, has control of the intervening communication channel, then E can compromise the communication in the following fashion without being detected:

1. A generates a public/private key pair {*PUa*, *PRa*} and transmits a message intended for B consisting of *PUa* and an identifier of A, *IDA*.
2. E intercepts the message, creates its own public/private key pair {*PUe*, *PRe*} and transmits *PUe*||*IDA* to B.
3. B generates a secret key, *Ks*, and transmits E(*PUe*, *Ks*).
4. E intercepts the message, and learns *Ks* by computing D(*PRe*, E(*PUe*, *Ks*)).
5. E transmits E(*PUa*, *Ks*) to A.

The result is that both A and B know *Ks* and are unaware that *Ks* has also been revealed to E. A and B can now exchange messages using *Ks* E no longer actively interferes with the communications channel but simply eavesdrops. Knowing *Ks* E can decrypt all messages, and both A and B are unaware of the problem. Thus, this simple protocol is only useful in an environment where the only threat is eavesdropping.

Secret Key Distribution with Confidentiality and Authentication

Figure 10.6, based on an approach suggested in [NEED78], provides protection against both active and passive attacks. We begin at a point when it is assumed that A and B have exchanged public keys by one of the schemes described earlier in this section. Then the following steps occur:

1. A uses B's public key to encrypt a message to B containing an identifier of A (*IDA*) and a nonce (*N*1), which is used to identify this transaction uniquely.

2. B sends a message to A encrypted with *PUa* 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 (1), the presence of *N*1 in message (2) assures A that the correspondent is B.
3. A returns *N*2 encrypted using B's public key, to assure B that its correspondent is A.
4. A selects a secret key *Ks* and sends *M* = E(*PUb*, E(*PRa*, *Ks*)) to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.
5. B computes D(*PUa*, D(*PRb*, *M*)) to recover the secret key.

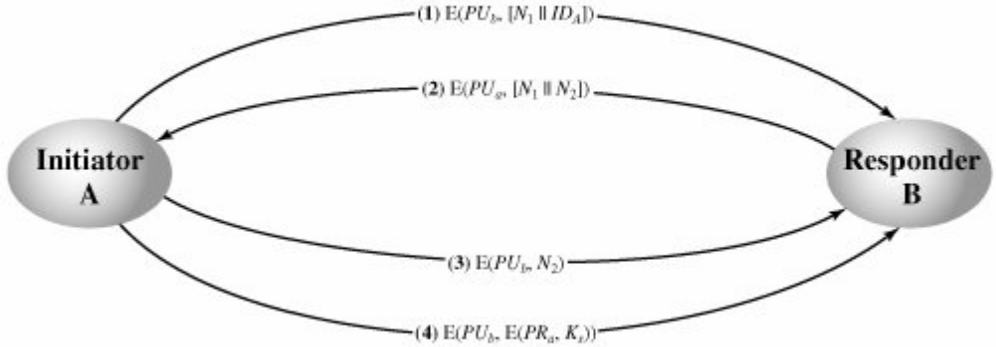


Figure 3.15 Public-Key Distribution of Secret Keys

Notice that the first three steps of this scheme are the same as the last three steps of Figure 10.3. The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

A Hybrid Scheme

Yet another way to use public-key encryption to distribute secret keys is a hybrid approach in use on IBM mainframes [LE93]. This scheme retains the use of a key distribution center (KDC) that shares a secret master 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 following rationale is provided for using this three-level approach:

● **Performance:** There are many applications, especially transaction-oriented applications, in which the session keys change frequently. Distribution of session keys by public-key encryption could degrade overall system performance because of the relatively high computational load of public-key encryption and decryption. With a three-level hierarchy, public-key encryption is used only occasionally to update the master key between a user and the KDC.

● **Backward compatibility:** The hybrid scheme is easily overlaid on an existing KDC scheme, with minimal disruption or software changes.

The addition of a public-key layer provides a secure, efficient means of distributing master keys. This is an advantage in a configuration in which a single KDC serves a widely distributed set of users.

POSSIBLE QUESTIONS

PART A (20 x 1 = 20 marks)

(ONLINE EXAMINATION)

PART B (5 x 6 = 30 marks)

1. Write the detailed description of RSA algorithm
2. Explain briefly about Diffie-Hellman key exchange
3. (i) Perform encryption/decryption using RSA algorithm for the following: p=3, q=11, e=7, m=5
4. Explain about public key cryptosystems with a neat diagram.
5. Describe in detail about the key distribution techniques and its hierarchy with neat diagram.
6. Explain about traffic confidentiality.

PART – C (1 x 10 = 10 marks)

1. Perform encryption and decryption using the RSA algorithm for the following:
   a. p=3; q=11, e=7; M=5
   b. p=5, q=11, e=3; M=9.
2. Users A and B use the Diffie-Hellman key exchange technique with a common prime q=71 and a primitive root $\alpha = 7$.
   a. If user A has private key X_A=5, what is A's public key Y_A?
   b. What is the shared secret key?
3. Describe in detail about the key distribution techniques

KARPAGAM ACADEMY OF HIGHER EDUCATION						
DEPARTMENT OF COMPUTER SCIENCE						
I M.Sc CS						
CRYPTOGRAPHY AND NETWORK SECURITY						
	UNIT 3					
S.NO	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	two major approaches to encryption placement _____ and	link and end to end	data	link	information	link and end to end
2	_____ are transmitted in the form of frames.	data	information	link	none	data
3	the _____ does not generate electromagnetic emanations and hence is not vulnerable to inductive taps.	fiber	feast	data	link	fiber
4	the most powerful and most common approach to securing the points of vulnerability highlighted in the preceding section is	data	decryption	encryption	link	encryption
5	_____ ,the encryption function is performed at a low level of the communications hierarchy.	link encryption	end to end	decryption	pre	link encryption
6	_____ model ,link encryption occurs at either the physical or link layers.	OSA	OSI	OSII	OSAI	OSI
7	_____,the encryption process is carried out at the two end systems.	data	link	end to end	information	end to end
8	the _____ accepts packets	FEA	FEP	FEK	FEPP	FEP
9	FEP stands for	front email processor	front end processor	form end processor	front end performing	front end processor
10	another concern related to traffic is the user of traffic patterns to create a	covert channel	distribution	link channel	data	covert channel
11	_____ a term refers to the means of delivering a key to two parties who wish to exchange data,without allowing others	form feed	key exchange	data processing	key distribution techniques	key distribution techniques
12	_____ used in electronic funds transfer and point of sale application.	PAN	file encryption	data	PIN	PIN
13	_____ key ,for encrypting files stored in publicly accessible locations.	file encryption	data encryption	PIN	file decryption	file encryption
14	_____ key,for general communication across a network.	Pin	file encryption	data encryption	file decryption	data encryption
15	A number of network security algorithms based on cryptography make use of .	random numbers	prime numbers	even numbers	odd numbers	random numbers
16	PRNGs stands for	Pseudorandom Number	Prime Number Generators	Pseudorandom Null Generators	PersonalNumber Generators	Pseudorandom Number
17	BBS	Blue Blue Shub Generator	Blum Blum Sub Generator	Blum Blum Shub Generator	Blue Blum Sub Generator	Blum Blum Shub Generator
18	a popular approach to generating secure pseudorandom number is known as the	BBS	BBA	BBC	BBD	BBS
19	CSPRBG	cylinder Secure Pseudorandom bit	cryptographically Secure	cylinder Secure Primebit	crypto Secure Prime bit	cryptographically Secure
20	A _____ number generator uses a nondeterministic source to produce randomness.	true random	false random	odd number	even number	true random
21	Asymmetric encryption transforms _____ into ciphertext using a one of two keys and an encryption algorithms.	cipher text	plaintext	covert	images	plaintext
22	the most widely used public key cryptosystem is ____	RSD	RAB	RSC	RSA	RSA
23	_____ is the readable message or data that is fed into the algorithms input	plaintext	cipher text	images	channel	plaintext
24	the _____ algorithm performs various transformation on the plaintext.	images	decryption	encryption	channel	encryption
25	_____ and _____ keys that have been selected so that if one is used for encryption , the other is used for decryption	public&protective	protected&private	encryption and decryption	public&private	public&private
26	_____ is the scrambled message produced as output	images	plaintext	ciphertext	channel	ciphertext
27	_____algorithm accepts the ciphertext and the matching key and produces the original plaintext	decryption	encryption	plaintext	images	decryption
28	the sender_____ a message with the recipients public key	encrypt	decrypt	plaintext	images	encrypt
29	the sender signs a message with its private key is known as	covert writing	analog signature	digital signature	type writer	digital signature
30	A_____ function is one that maps a domain into a range such that every function value has a unique inverse	one-way	multiway	twoway	three way	one-way
31	plaintext is encrypted in blocks,with each block having a binary value less than some number and the block size must be less	log2(n)	logn	log n*n	logn2	log2(n)
32	_____ involves trying all possible kays	poly force	hill force	brute force	none	brute force
33	_____ depends on the running time of the decryption algorithm	one pad	timing attack	signature	force	timing attack
34	_____types of attack exploits properties of the RSA algorithm	chosen plain text	encryption	chosen ciphertext attack	decryption	chosen ciphertext attack
35	_____ are several approaches,all equivalent in effort to factoring the products of two primes	timing attack	force attack	burst force	mathematical attacks	mathematical attacks
36	SNFS stands for	spell number fiels sieve	special number fiels sieve	special null fiels sieve	special number fiels store	special number fiels sieve
37	GNFS stands for	generalized number field store	generalized null field sieve	generalized number field	great number field sieve	generalized number field
38	_____multiply the ciphertext by a random number before performing exponentiation	blinding	random delay	constant	none	blinding
39	better performance could be achieved by adding a _____ to the exponentiation algorithm to confuse the timing attack.	random delay	blinding	constant	none	random delay
40	a simple public key algorithm is _____ key exchange	brute force	hellman	diffie-hellman	none	diffie-hellman
41	ECC stands for	Elliptic Curve Cryptography	elliptic corner cryptography	End Curve Cryptography	Elliptic Cursor Cryptography	Elliptic Curve Cryptography
42	the _____ algorithm depends for its effectiveness on the difficulty of computing discrete logarithmss	hellman	diffie-hellman	brust force	none	diffie-hellman

43	the protocol depicted insecure against an adversary who can intercept messages and then either relay the intercepted	man in the middle	man in the front	man in the end	none	man in the middle
44	_____,the hybrid schema is easily overlaid on an existing KDC schema,with minimal disruption or software changes	forward compatibility	middle compatibility	backward compatibility	front compatibility	backward compatibility
45	The _____ attack can endanger the security of the Diffie-Hellman method if two parties are not authenticated to each	man-in-the-middle	ciphertext attack	plaintext attack	none of the above	man-in-the-middle
46	The _____ method provides a one-time session key for two parties.	Diffie-Hellman	RSA	DES	AES	Diffie-Hellman
47	____possible approaches are there to attack the RSA Algorithm.	5	4	3	6	4
48	____approach is used to attack the RSA Algorithm.	Random	Modulus	Bruteforce	Factorial	Bruteforce
49	The process of verifying an identity for a system entity _____	AES	authentication	reliability	security	authentication
50	CCA	cost compare attack	chosen cipher attack	chosen cost attack	cost compare AES	chosen cipher attack
51	OAEP	Optimal asymmetric	Optical AEP	Optical and encryption	Optimal asymmetric end	Optimal asymmetric
52	Message digest uses_____function	Heap	Hash	Math	logn2	Hash
53	A sequence of 4 bits is _____	byte	octet	nibble	binary	nibble
54	Program that consumes system resources by replicating itself	Recursion	Factorial	Bacteria	Fibonacci	Bacteria
55	_____ is a technique which is not use encryption for secure the message.	Cryptology	cryptography	steganography	cipher	steganography
56	steganography means	uncovered writing	covered writing	encryption	decryption	covered writing
57	caesar cipher uses	C=E(3,p)=(p+3)mod 26	C=E(4,p)=(p+3)mod 26	C=E(3,p)=(p+5)mod 26	C=E(3,p)=(p+6)mod 26	C=E(3,p)=(p+3)mod 26
58	_____technique used in steganography.	caesar cipher	playfair cipher	Hill cipher	Invisible ink	Invisible ink
59	In asymmetric key cryptography, the private key is kept by	sender	receiver	both	none	receiver
60	Which one of the following algorithm is not used in asymmetric-key cryptography?	RSA algorithm	diffie-hellman algorithm	electronic code book algorithm	none	electronic code book algorithm

UNIT- IV

Syllabus

Message Authentication and hash functions – Authentication Functions – Message Authentication Codes (MAC's) Functions – Security of Hash Functions and MAC's Digital Signatures and Authentication Protocols – Digital Signatures – Digital Signature Standard

AUTHENTICATION AND HASH FUNCTION

Message Authentication and hash functions

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.

● Symmetric encryption provides authentication among those who share the secret key. Encryption of a message by a sender's private key also provides a form of authentication.

● The two most common cryptographic techniques for message authentication are a message authentication code (MAC) and a secure hash function.

● A MAC is an algorithm that requires the use of a secret key. A MAC takes a variable length message and a secret key as input and produces an authentication code. A recipient in possession of the secret key can generate an authentication code to verify the integrity of the message.

● A hash function maps a variable-length message into a fixed length hash value, or message digest. For message authentication, a secure hash function must be combined in some fashion with a secret key.

AUTHENTICATION REQUIREMENTS

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

- **Disclosure** – releases of message contents to any person or process not possessing the appropriate cryptographic key.
- **Traffic analysis** – discovery of the pattern of traffic between parties.
- **Masquerade** – insertion of messages into the network fraudulent source.
- **Content modification** – changes to the content of the message, including insertion deletion, transposition and modification.
- **Sequence modification** – any modification to a sequence of messages between parties, including insertion, deletion and reordering.
- **Timing modification** – delay or replay of messages.
- **Source repudiation** – denial of transmission of message by source.
- **Destination repudiation** – denial of transmission of message by destination.

Measures to deal with first two attacks are in the realm of message confidentiality. Measures to deal with 3 through 6 are regarded as message authentication. Item 7 comes under digital signature and dealing with item 8 may require a combination of digital signature and a protocol to counter this attack.

AUTHENTICATION FUNCTIONS

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

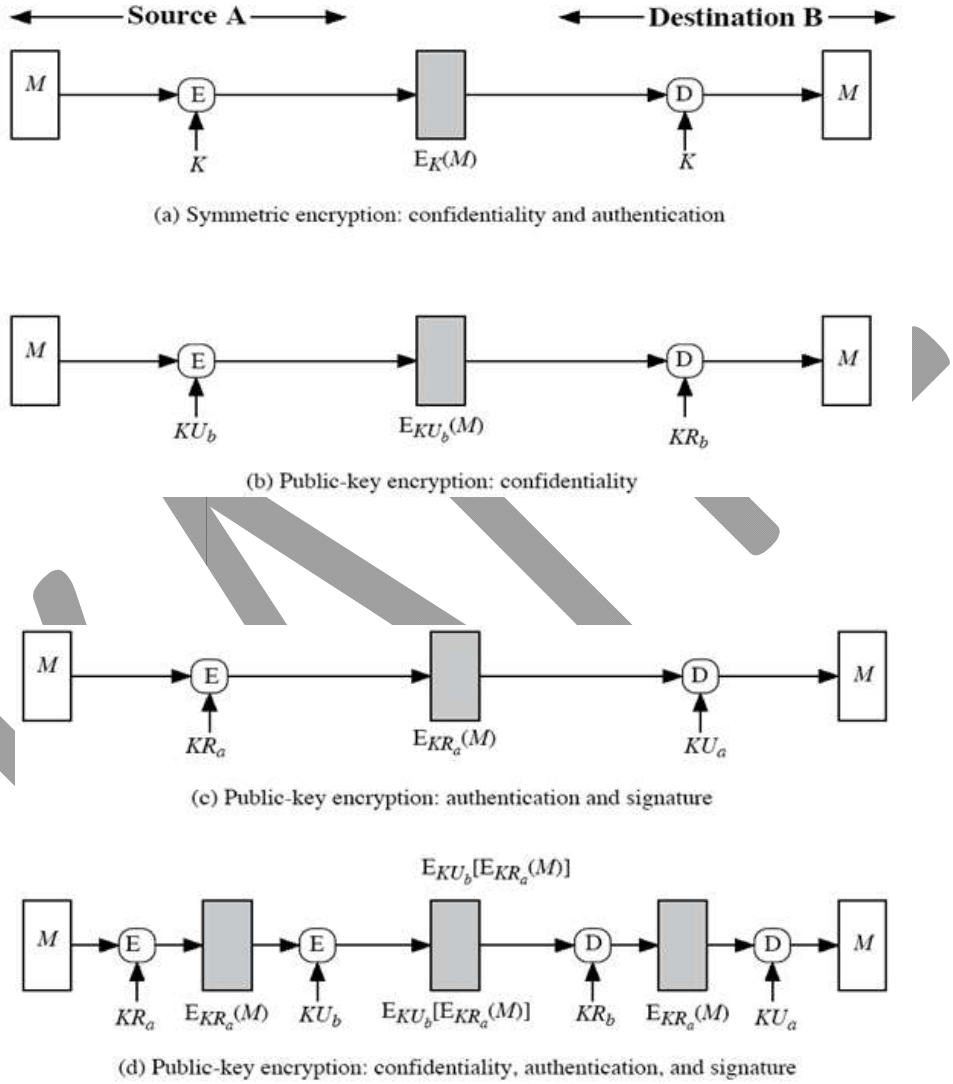
The different types of functions that may be used to produce an authenticator are as follows:

- **Message encryption** – the cipher text of the entire message serves as its authenticator.
- **Message authentication code (MAC)** – a public function of the message and a secret key that produces a fixed length value serves as the authenticator.

- **Hash function** – a public function that maps a message of any length into a fixed length hash value, which serves as the authenticator.

Message encryption

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



(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 4.1 Basic Uses of Message Encryption

Suppose the message can be any arbitrary bit pattern. In that case, there is no way to determine automatically, at the destination whether an incoming message is the ciphertext of a legitimate message. One solution to this problem is to force the plaintext to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption

function. We could, for example, append an error detecting code, also known as Frame Check Sequence (FCS) or checksum to each message before encryption.

'A' prepares a plaintext message M and then provides this as input to a function F that produces an FCS. The FCS is appended to M and the entire block is then encrypted. At the destination, B decrypts the incoming block and treats the result as a message with an appended FCS. B applies the same function F to attempt to reproduce the FCS. If the calculated FCS is equal to the incoming FCS, then the message is considered authentic.

In the internal error control, the function F is applied to the plaintext, whereas in external error control, F is applied to the ciphertext (encrypted message).

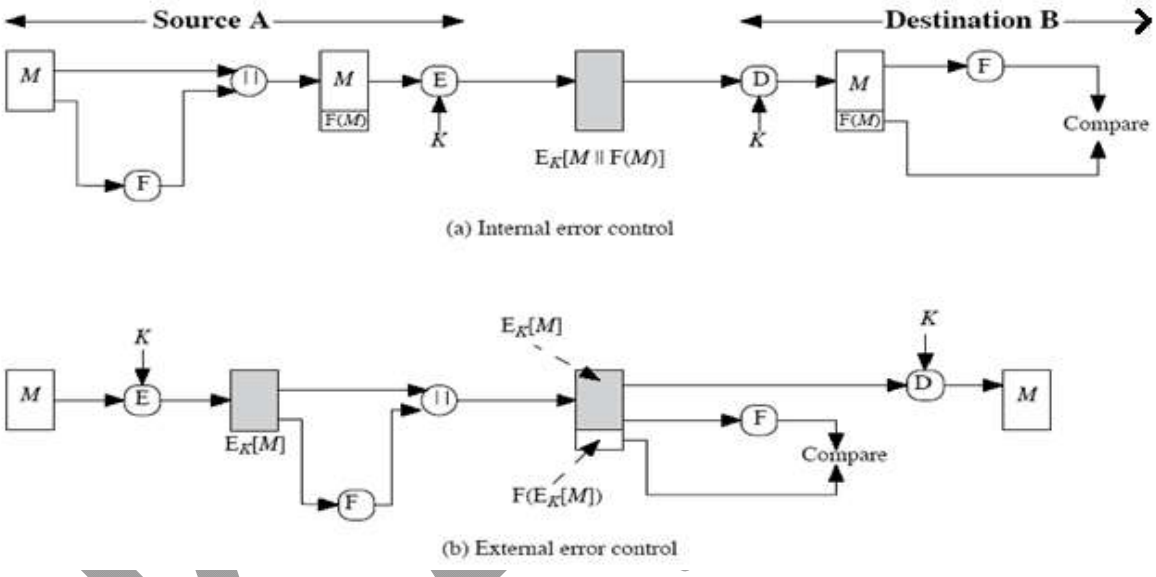


Figure 4.2 Internal and External Error Control

MESSAGE AUTHENTICATION CODE (MAC)

An alternative authentication technique involves the use of secret key to generate a small fixed size block of data, known as cryptographic checksum or MAC that is appended to the message. This technique assumes that two communication parties say A and B, share a common secret key 'k'. When A has to send a message to B, it calculates the MAC as a function of the message and the key.

$MAC = C_K(M)$ 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 shared secret key, to generate a new MAC. The received MAC is compared to the calculated MAC. If it is equal, then the message is considered authentic.

A MAC function is similar to encryption. One difference is that MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many-to-one function.

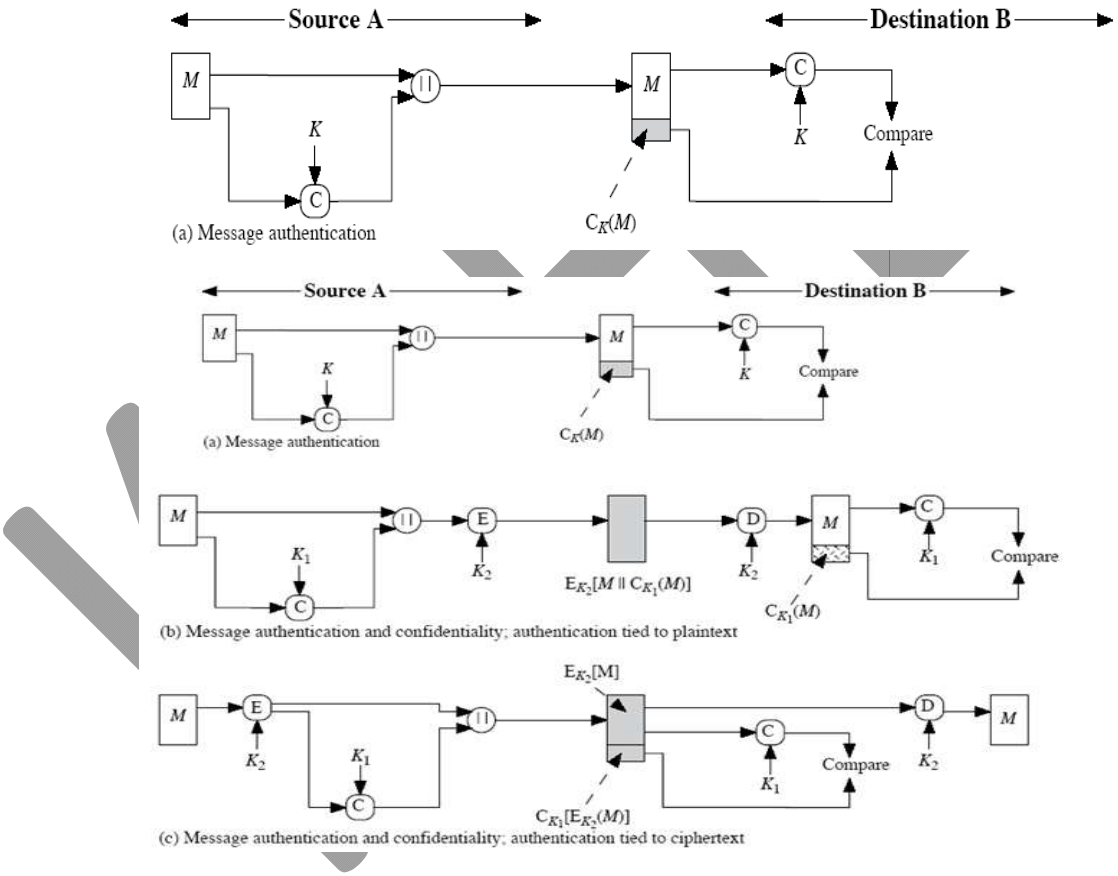


Figure 4.3: Basic Uses of Message Authentication Code (MAC)

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 received MAC is compared to the calculated MAC (Figure 3.4a). 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. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.

3. If the message includes a sequence number (such as is used with HDLC, X.25, and TCP), then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

A MAC function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many-to-one function. The domain of the function consists of messages of some arbitrary length, whereas the range consists of all possible MACs and all possible keys. If an n-bit MAC is used, then there are 2^n possible MACs, whereas there are N possible messages with $N >> 2^n$. Furthermore, with a k-bit key, there are $2k$ possible keys.

The process depicted in Figure 3.4a provides authentication but not confidentiality, because the message as a whole is transmitted in the clear. Confidentiality can be provided by performing message encryption either after (Figure 3.4b) or before (Figure 3.4c) the MAC algorithm.

In both these cases, two separate keys are needed, each of which is shared by the sender and the receiver. In the first case, the MAC is calculated with the message as input and is then concatenated to the message. The entire block is then encrypted. In the second case, the message is encrypted first. Then the MAC is calculated using the resulting ciphertext and is concatenated to the ciphertext to form the transmitted block. Typically, it is preferable to tie the authentication directly to the plaintext, so the method of Figure 3.4b is used.

Finally, note that the MAC does not provide a digital signature because both sender and receiver share the same key.

HASH FUNCTIONS

A variation on the message authentication code is the one way hash function. As with MAC, a hash function accepts a variable size message M as input and produces a fixed-size output, referred to as hash code H(M). Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a message digest or hash value.

There are varieties of ways in which a hash code can be used to provide message authentication, as follows:

a) The message plus the hash code is encrypted using symmetric encryption. This is identical to that of internal error control strategy. Because encryption is applied to the entire message plus the 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.

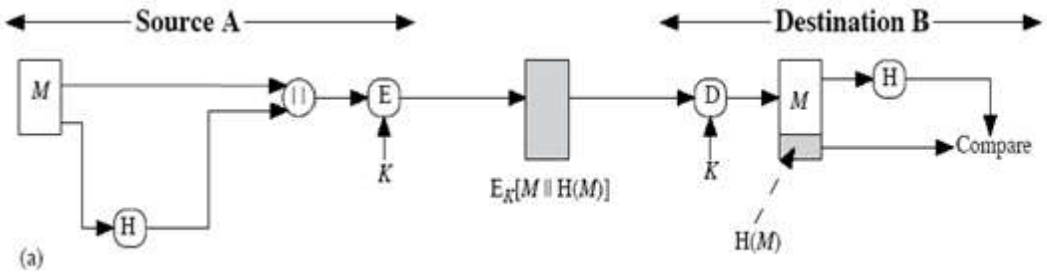
c) Only the hash code is encrypted, using the public key encryption and using the

sender's private key. It provides authentication plus the digital signature.

d) If confidentiality as well as digital signature is desired, then the message plus the public key encrypted hash code can be encrypted using a symmetric secret key.

e) This technique uses a hash function, but no encryption for message authentication. This technique assumes that the two communicating parties share a common secret value 'S'. The source computes the hash value over the concatenation of M and S and appends the resulting hash value to M.

f) Confidentiality can be added to the previous approach by encrypting the entire message plus the hash code.



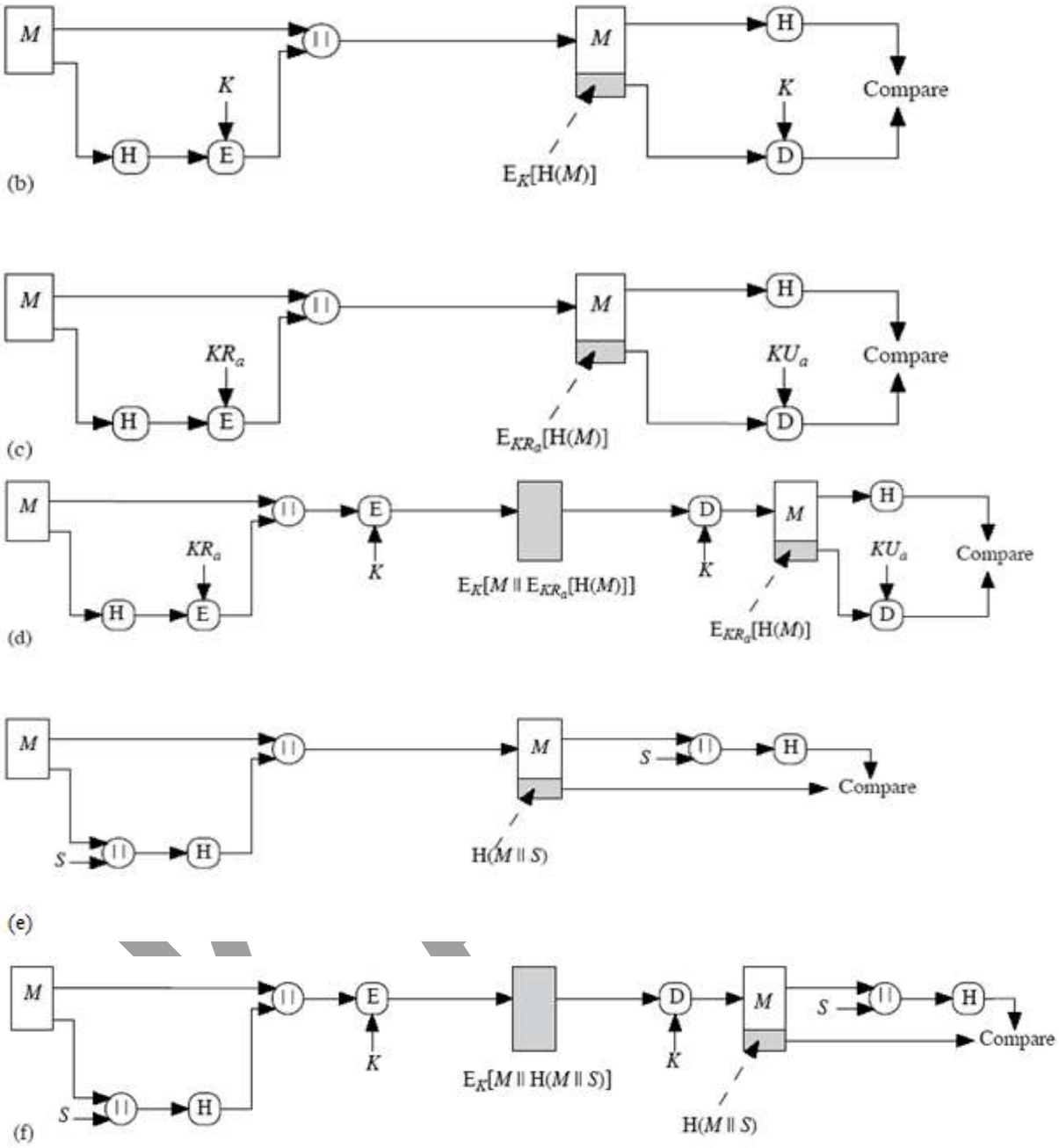


Figure 4.4. Basic Uses of Hash Function

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

$$h = H(M)$$

where M is a variable-length message and H(M) is the fixed-length hash value. The hash value is

appended to the message at the source at a time when the message is assumed orknown to be correct. The receiver authenticates that message by recomputing the hashvalue.

When confidentiality is not required, methods (b) and (c) have an advantage over those that encrypt the entire message in that less computation is required.

Requirements for a Hash Function

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the one-way property.
5. For any given block x, it is computationally infeasible to find $y \neq x$ such that H(y) = H(x). This is sometimes referred to as **weak collision resistance**.

6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as **strong collision resistance**.

The first three properties are requirements for the practical application of a hash function to message authentication.

The fourth property, the one-way property, states that it is easy to generate a code given a message but virtually impossible to generate a message given a code. The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found. This prevents forgery when an encrypted hash code is used (Figures b and c).

The sixth property refers to how resistant the hash function is to a type of attack known as the birthday attack, which we examine shortly.

Simple Hash Functions

All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of n-bit blocks. The input is processed one block at a time in an iterative fashion to produce an n-bit hash function.

One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as follows:

$C_i = b_{i1} \oplus b_{i1} \oplus \ldots \oplus b_{im}$

where

C_i = ith bit of the hash code, $1 \leq i \leq n$

m = number of n-bit blocks in the input b_{ij} = ith bit in jth block

$\oplus$ = XOR operation

Thus, the probability that a data error will result in an unchanged hash value is 2^n. With more predictably formatted data, the function is less effective. For example, in most normal text files, the high-order bit of each octet is always zero. So if a 128-bit hash value is used, instead of an effectiveness of 2^{128}, the hash function on this type of data has an effectiveness of 2^{112}.

A simple way to improve matters is to perform a one-bit circular shift, or rotation, on the hash value after each block is processed. The procedure can be summarized as follows:

1. Initially set the n-bit hash value to zero.
2. Process each successive n-bit block of data as follows:
   a. Rotate the current hash value to the left by one bit.

b. XOR the block into the hash value.

Birthday Attacks

Suppose that a 64-bit hash code is used. One might think that this is quite secure. For example, if an encrypted hash code C is transmitted with the corresponding unencrypted message M, then an opponent would need to find an M' such that H(M') = H(M) to substitute another message and fool the receiver.

On average, the opponent would have to try about 2^{63} messages to find one that matches the hash code of the intercepted message

However, a different sort of attack is possible, based on **the birthday paradox** The source, A, is prepared to "sign" a message by appending the appropriate m-bit hash code and encrypting that hash code with A's private key (Figure 3.7c).

1. The opponent generates $2^{m/2}$ variations on the message, all of which convey essentially the same meaning. (fraudulent message
2. The two sets of messages are compared to find a pair of messages that produces the same hash code. The probability of success, by the birthday paradox, is greater than 0.5. If no match is found, additional valid and fraudulent messages are generated until a match is made.
3. The opponent offers the valid variation to A for signature. This signature can then be attached to the fraudulent variation for transmission to the intended recipient. Because the two variations have the same hash code, they will produce the same signature; the opponent is assured of success even though the encryption key is not known.

Thus, if a 64-bit hash code is used, the level of effort required is only on the order of 2^{32} .

Block Chaining Techniques

Divide a message M into fixed-size blocks $M_1, M_2, ..., M_N$ and use a symmetric encryption system such as DES to compute the hash code G as follows:

H_o = initial value $H_i = E_{Mi} [H_{i-1}]$ $G = H_N$

This is similar to the CBC technique, but in this case there is no secret key. As with any hash code, this scheme is subject to the birthday attack, and if the encryption algorithm is DES and only a 64-bit hash code is produced, then the system is vulnerable.

Furthermore, another version of the birthday attack can be used even if the opponent has access to only one message and its valid signature and cannot obtain multiple signings.

Here is the scenario; we assume that the opponent intercepts a message with a signature in the form of an encrypted hash code and that the unencrypted hash code is m bits long:

1. Use the algorithm defined at the beginning of this subsection to calculate the unencrypted hash code G.
2. Construct any desired message in the form $Q_1, Q_2, ..., Q_{N2}$. 3. Compute for $H_i = EQ_i [H_{i-1}]$ for $1 \leq i \leq (N-2)$.
4. Generate $2^{m/2}$ random blocks; for each block X, compute $E_X[H_{N-2}.]$ Generate an additional $2^{m/2}$ random blocks; for each block Y, compute $D_Y[G]$, where D is the decryption function corresponding to E.
5. Based on the birthday paradox, with high probability there will be an X and Y such that $E_X [H_{N-2}] = D_Y[G]$.
6. Form the message $Q_1, Q_2, ..., Q_{N-2}, X, Y$. This message has the hash code G and therefore can be used with the intercepted encrypted signature.

This form of attack is known as a **meet-in-the-middle attack**.

Message Authentication Codes

A MAC, also known as a cryptographic checksum, is generated by a function C of the form MAC = C(*K*, *M*) where *M* is a variable-length message, *K* is a secret key shared only by sender and receiver, and C(*K, M*) is the fixed-length authenticator. The MAC 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 recomputing the MAC.

Requirements for MAC:

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key. Barring some weakness in the algorithm, the opponent must resort to a brute-force attack using all possible keys. On average, such an attack will require $2^{(k-1)}$ attempts for a k-bit key.

In the case of a MAC, the considerations are entirely different. Using brute-force methods, how would an opponent attempt to discover a key?

If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose k > n; that is, suppose that the key size is greater than the MAC size. Then, given a known M_1 and MAC_1, with $MAC_1 = CK(M_1)$, the cryptanalyst can perform $MAC_i = CK_i(M_1)$ for all possible key values K_i.

At least one key is guaranteed to produce a match of $MAC_i = MAC_1$.

Note that a total of 2^k MACs will be produced, but there are only $2^n < 2^k$ different MAC values. Thus, a number of keys will produce the correct MAC and the opponent has no way of knowing which is the correct key. On average, a total of $2^k/2^n = 2^{(k-n)}$ keys will produce a match. Thus, the opponent must iterate the attack:

- **Round 1**

  Given: M_1, $MAC_1 = CK(M_1)$

  Compute $MAC_i = CK_i(M_1)$ for all 2^k keys

  Number of matches $\approx 2^{(k-n)}$

- **Round 2**

Given: M_2, $MAC_2 = CK(M_2)$

Compute $MAC_i = CK_i(M_2)$ for the $2^{(k-n)}$ keys resulting from Round 1

Number of matches $\approx 2^{(k-2xn)}$

and so on. On average, a rounds will be needed if k = a x n. For example, if an 80-bit key is used and the MAC is 32 bits long, then the first round will produce about 2^{48} possible keys. The second round will narrow the possible keys to about 2^{16} possibilities. The third round should produce only a single key, which must be the one used by the sender.

If the key length is less than or equal to the MAC length, then it is likely that a first round will produce a single match.

Thus, a brute-force attempt to discover the authentication key is no less effort and may be more effort than that required to discover a decryption key of the same length. However, other attacks that do not require the discovery of the key are possible.

Consider the following MAC algorithm. Let $M = (X_1||X_2||...||X_m)$ be a message that is treated as a concatenation of 64-bit blocks X_i. Then define

$$\Delta(M) = X1 \oplus X2 \oplus \ldots \oplus Xm$$

$$Ck(M) = Ek(\Delta(M))$$

where $\oplus$ is the exclusive-OR (XOR) operation and the encryption algorithm is DES in electronic codebook mode. Thus, the key length is 56 bits and the MAC length is 64 bits. If an opponent observes $\{M||C(K, M)\}$, a brute-force attempt to determine K will require at least 2^{56} encryptions. But the opponent can attack the system by replacing X_1 through X_{m-1} with any desired values Y_1 through Y_{m-1} and replacing X_m with Y_m where Y_m is calculated as follows:

$Y_m = Y_1 \oplus Y_2 \oplus \ldots \oplus Y_{m-1} \oplus \Delta(M)$

The opponent can now concatenate the new message, which consists of Y_1 through Y_m, with the original MAC to form a message that will be accepted as authentic by the receiver. With this tactic, any message of length 64 x (m-1) bits can be fraudulently inserted.

Then the MAC function should satisfy the following requirements:

The MAC function should have the following properties:

- If an opponent observes M and $C_K(M)$, it should be computationally infeasible for the opponent to construct a message M' such that $C_K(M') = C_K(M)$
- $C_K(M)$ should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that $C_K(M) = C_K(M')$ is 2^{-n} where n is the number of bits in the MAC.
- Let M' be equal to some known transformation on M. i.e., M' = f(M).

MAC based on DES

One of the most widely used MACs, referred to as Data Authentication Algorithm (DAA) is based on DES.

The algorithm can be defined as using cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero. The data to be authenticated are grouped into contiguous 64-bit blocks: D_1, D_2 … D_n. if necessary, the final block is padded on the right with zeros to form a full 64-bit block. Using the DES encryption algorithm and a secret key, a data authentication code (DAC) is calculated as follows:

$$O_1 = E_K(D_1)$$

$$O_2 = E_K(D_2 \oplus O_1)$$

$$O_3 = E_K(D_3 \oplus O_2) \ldots .$$

$O_N = E_K(D_N \oplus O_{N-1})$

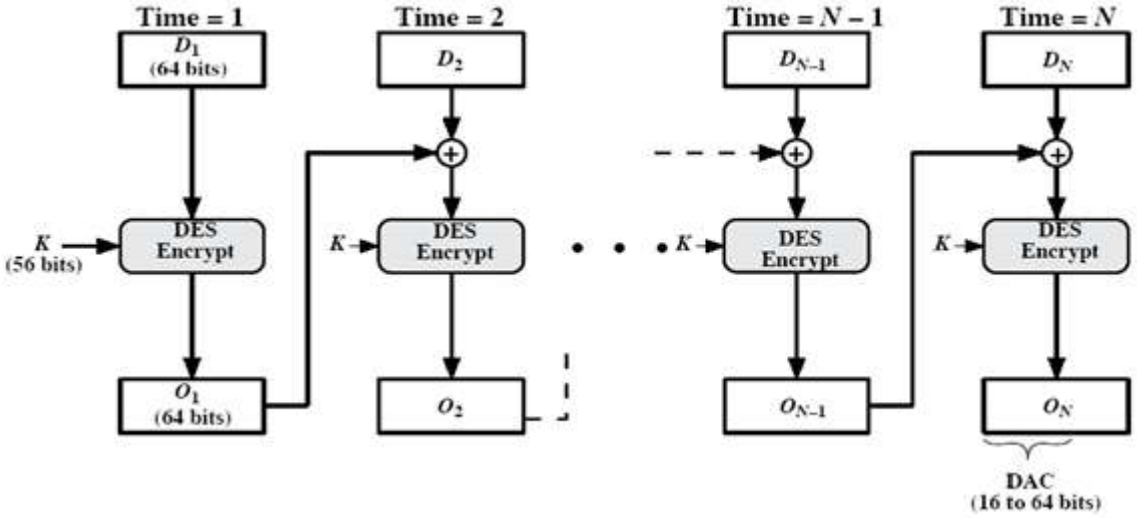


Figure 4.5: Data Authentication Algorithm (FIPS PUB 113)

The DAC consists of either the entire block *ON* or the leftmost *M* bits of the block, with 16 <= *M* <= 64.

SECURITY OF HASH FUNCTIONS AND 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.

i. Brute-Force Attacks

The nature of brute-force attacks differs somewhat for hash functions and MACs.

Hash Functions

The strength of a hash function against brute-force attacks depends solely on the length of the hash code produced by the algorithm. Recall from our discussion of hash functions that there are three desirable properties:

- One-way: For any given code h, it is computationally infeasible to find x such that H(x) = h.
- Weak collision resistance: For any given block x, it is computationally infeasible to find y x

with H(y) = H(x).

Strong collision resistance: It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

For a hash code of length n, the level of effort required, as we have seen is proportional to the following:

One way	2^n
Weak collision resistance	2^n
Strong collision resistance	$2^{n/2}$

Message Authentication Codes

A brute-force attack on a MAC is a more difficult undertaking because it requires known message-MAC pairs.. To attack a hash code, we can proceed in the following way. Given a fixed message x with n-bit hash code h = H(x), a brute-force method of finding a collision is to pick a random bit string y and check if H(y) = H(x). The attacker can do this repeatedly off line. To proceed, we need to state the desired security property of a MAC algorithm, which can be expressed as follows:

Computation resistance: Given one or more text-MAC pairs (x_i, $C_K[x_i]$), it is computationally infeasible to compute any text-MAC pair (x, C_K(x)) for any new input $x \neq x_i$.

In other words, the attacker would like to come up with the valid MAC code for a given message x. There are two lines of attack possible: Attack the key space and attack the MAC value. We examine each of these in turn.

To summarize, the level of effort for brute-force attack on a MAC algorithm can be expressed as min(2^k, 2^n). The assessment of strength is similar to that for 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.

ii. 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.

Hash Functions

In recent years, there has been considerable effort, and some successes, in developing cryptanalytic attacks on hash functions. To understand these, we need to look at the overall structure of a typical secure hash function, and is the structure of most hash functions in use today, including SHA and Whirlpool.

The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each. If necessary, the final block is padded to b bits. The final block also includes the value of the total length of the input to the hash function.

The inclusion of the length makes the job of the opponent more difficult. Either the opponent must find two messages of equal length that hash to the same value or two messages of differing lengths that, together with their length values, hash to the same value.

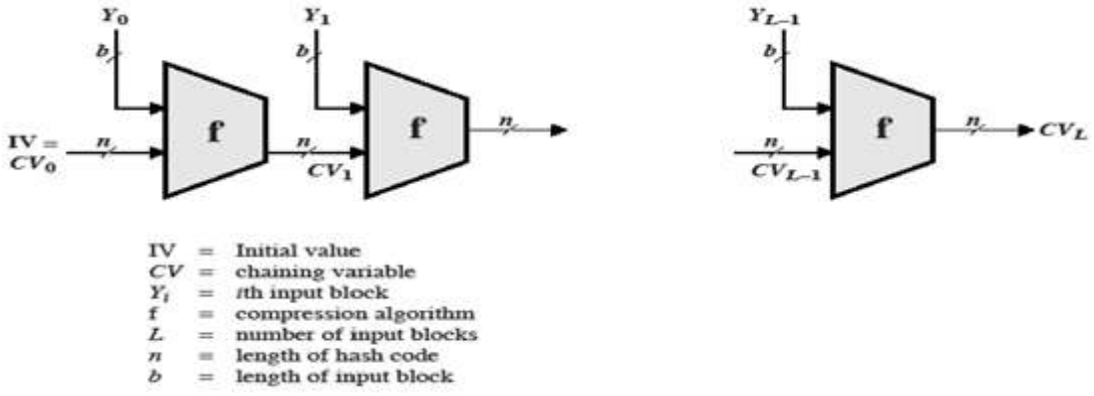


Figure 4.6: General Structure of Secure Hash Code

The hash algorithm involves repeated use of a **compression function**, f, that takes two inputs (an n-bit input from the previous step, called the chaining variable, and a b-

bit block) and produces an n-bit output. At the start of hashing, the chaining variable has an initial value that is specified as part of the algorithm. The final value of the chaining variable is the hash value. Often, b > n; hence the term compression. The hash function can be summarized as follows:

CV_0 = IV = initial n-bit value $CV_i = f(CV_{i-1}, Y_{i-1})$ $1 \leq i \leq L$ $H(M) = CV_L$

where the input to the hash function is a message M consisting of the blocks $Y_0, Y_1, ..., Y_{L-1}$. The structure can be used to produce a secure hash function to operate on a message of any length.

Message Authentication Codes

There is much more variety in the structure of MACs than in hash functions, so it is difficult to generalize about the cryptanalysis of MACs. Further, far less work has been done on developing such attacks.

Digital Signatures and Authentication Protocols

A digital signature is an authentication mechanism that enables the creator of a message to attach a code that acts as a signature. 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 and integrity of the message.

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

The most important development from the work on public-key cryptography is the digital signature. The digital signature provides a set of security capabilities that would be difficult to implement in any other way. We begin this chapter with an overview of digital signatures. Then we look at authentication protocols, many of which depend on the use of the digital signature. Finally, we introduce the Digital Signature Standard (DSS).

Digital Signatures

Requirements

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

For example, suppose that John sends an authenticated message to Mary, using one of the schemes of Figure 3.4. Consider the following disputes that could arise:

1. Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.
2. John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

Both scenarios are of legitimate concern. Here is an example of the first scenario: An electronic funds transfer takes place, and the receiver increases the amount of funds transferred and claims that the larger amount had arrived from the sender. An example of the second scenario is that an electronic mail message contains instructions to a stockbroker for a transaction that subsequently turns out badly. The sender pretends that the message was never sent. In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature. The digital signature is analogous to the handwritten signature.

It must have the following properties:

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

Thus, the digital signature function includes the authentication function.

On the basis of these properties, we can formulate the following requirements for a digital signature**:**

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information unique to the sender, to prevent both forgery and denial.
- It must be relatively easy to produce the digital signature.
- It must be relatively easy to recognize and verify the digital signature.
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
- It must be practical to retain a copy of the digital signature in storage.

A secure hash function, embedded in a scheme such as that of Figure 11.5c or d, satisfies these requirements.

A variety of approaches has been proposed for the digital signature function. These approaches fall into two categories: direct and arbitrated.

Direct Digital Signature

The direct digital signature involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source. A digital signature may be formed by encrypting the entire message with the sender's private key or by encrypting a hash code of the message with the sender's private key. Confidentiality can be provided by further encrypting the entire message plus signature with either the receiver's public key (public-key encryption) or a shared secret key (symmetric encryption).

One example is to require every signed message to include a timestamp (date and time) and to require prompt reporting of compromised keys to a central authority. Another threat is that some private key might actually be stolen from X at time T. The opponent can then send a message signed with X's signature and stamped with a time before or equal to T.

Arbitrated Digital Signature

The problems associated with direct digital signatures can be addressed by using an arbiter. As with direct signature schemes, there is a variety of arbitrated signature schemes. In general terms,

they all operate as follows.

Every signed message from a sender X to a receiver Y goes first to an arbiter A, who subjects the message and its signature to a number of tests to check its origin and content. The message is then dated and sent to Y with an indication that it has been verified to the satisfaction of the arbiter.

The presence of A solves the problem faced by direct signature schemes: that X might disown the message. The arbiter plays a sensitive and crucial role in this sort of scheme, and all parties must have a great deal of trust that the arbitration mechanism is working properly. The use of a trusted system, described in Chapter 20, might satisfy this requirement.

Digital Signature Standard

The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Standard (DSS).

The DSS makes use of the Secure Hash Algorithm (SHA) described in Chapter 12 and presents a new digital signature technique, the Digital Signature Algorithm (DSA).

The DSS was originally proposed in 1991 and revised in 1993 in response to public feedback concerning the security of the scheme.

There was a further minor revision in 1996. In 2000, an expanded version of the standard was issued as FIPS 186-2.

This latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography. In this section, we discuss the original DSS algorithm.

The DSS Approach

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 contrasts the DSS approach for generating digital signatures to that used with RSA. 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.

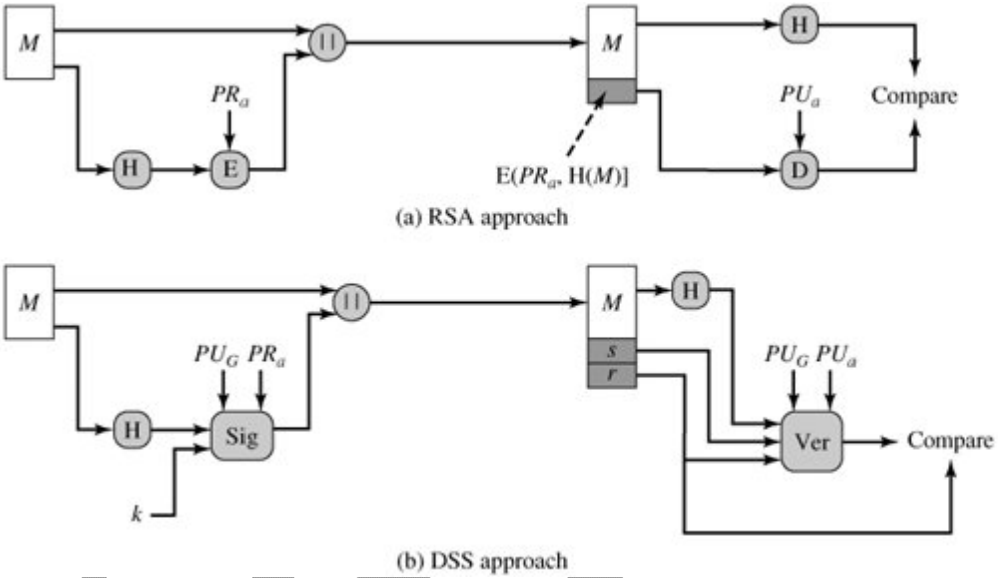


Fig 4.7 Two Approaches to Digital Signatures

The DSS approach also makes use of a hash function. The hash code is provided as input to a signature function along with a random number *k* generated for this particular signature. The signature function also depends on the sender's private key (*PRa*)and a set of parameters known to a group of communicating principals. We can consider this set to constitute a global public key (*PUG*). [4]

The result is a signature consisting of two components, labeled *s* and *r*. At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender's public key (*PUa*), which is paired with the sender's private key.

The output of the verification function is a value that is equal to the signature component *r* if the signature is valid. The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

The Digital Signature Algorithm

The DSA is based on the difficulty of computing discrete logarithms (see Chapter 8) and is based on schemes originally presented by ElGamal [ELGA85] and Schnorr

Figure 13.2 summarizes the algorithm.

There are three parameters that are public and can be common to a group of users. 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.

Global Public-Key Components	
p	prime number where $2^{L-1} < p < 2^L$ for $512 \leq L \leq 1024$ and L a multiple of 64; i.e., bit length of between 512 and 1024 bits in increments of 64 bits
q	prime divisor of $(p-1)$, where $2^{159} < q < 2^{160}$; i.e., bit length of 160 bits
g	$= h^{(p-1)/q} \bmod p$, where h is any integer with $1 < h < (p-1)$ such that $h^{(p-1)/q} \bmod p > 1$
User's Private Key	
x	random or pseudorandom integer with $0 < x < q$
User's Public Key	
y	$= g^x \bmod p$
User's Per-Message Secret Number	
k	= random or pseudorandom integer with $0 < k < q$
Signing	
r	$= (g^k \bmod p) \bmod q$
s	$= [k^{-1}(H(M) + xr)] \bmod q$
Signature = (r, s)	
Verifying	
w	$= (s')^{-1} \bmod q$
u1	$= [H(M')w] \bmod q$
u2	$=(r')w \bmod q$
v	$= [(g^{u1} y^{u2}) \bmod p] \bmod q$
TEST: $v = r'$	
M	= message to be signed
$H(M)$	= hash of M using SHA-1
M', r', s'	= received versions of M, r, s

- With these numbers in hand, each user selects a private key and generates a public key.
- The private key x must be a number from 1 to $(q-1)$ and should be chosen randomly or pseudorandomly.
- The public key is calculated from the private key as $y = gx \bmod p$.
- The calculation of y given x is relatively straightforward. However, given the public key y, it is believed to be computationally infeasible to determine x, which is the discrete logarithm of y to the base g, mod p (see Chapter 8).

- To create a signature, a user calculates two quantities, *r* and *s*, that are functions of the public key components (*p, q, g*), the user's private key (*x*), the hash code of the message, H(*M*), and an additional integer *k* that should be generated randomly or pseudorandomly and be unique for each signing.
- At the receiving end, verification is performed using the formulas shown in Figure 13.2.
- The receiver generates a quantity *v* that is a function of the public key components, the sender's public key, and the hash code of the incoming message. If this quantity matches the *r* component of the signature, then the signature is validated.

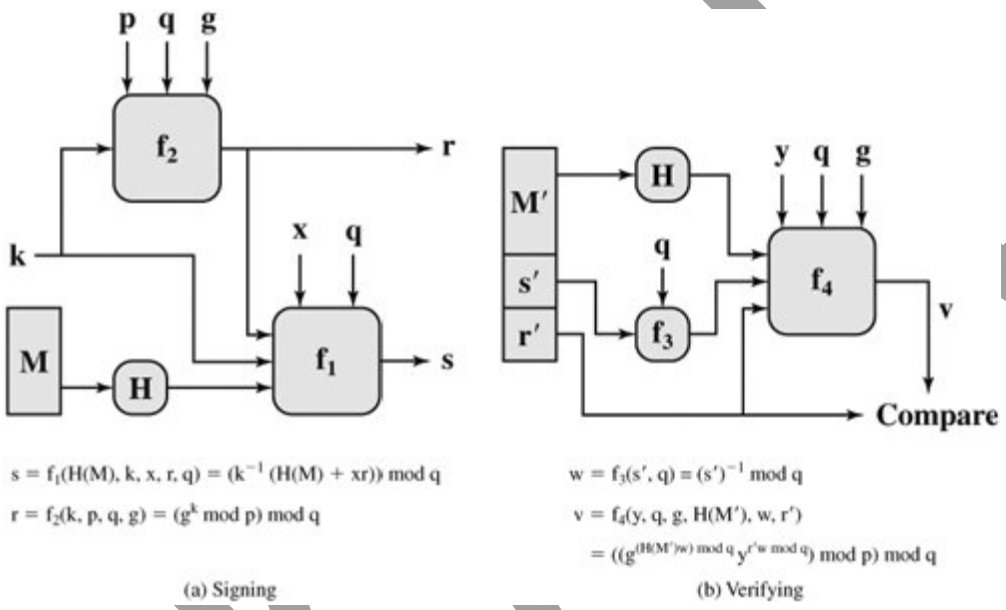


Fig 4.8. DSS Signing and Verifying

The structure of the algorithm, as revealed in Figure, 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.

Authentication Protocols

In this section, we focus on two general areas (mutual authentication and one-way authentication) and examine some of the implications of authentication techniques in both.

Mutual Authentication

● Mutual authentication protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys.

● In one-way authentication, the recipient wants some assurance that a message is from the alleged sender.

An important application area is that of mutual authentication protocols. Such protocols enable communicating parties to satisfy themselves mutually about each other's identity and to exchange session keys. This topic was examined in Section 7.3 (symmetric techniques) and Section 10.1 (publickey techniques). There, the focus was key distribution. We return to this topic here to consider the wider implications of authentication.

Central to the problem of authenticated key exchange are two issues: confidentiality and timeliness.

To prevent masquerade and to prevent compromise of session keys, essential identification and session key information must be communicated in encrypted form. This requires the prior existence of secret or public keys that can be used for this purpose.

The second issue, timeliness, is important because of the threat of message replays. Such replays, at worst, could allow an opponent to compromise a session key or successfully impersonate another party. At minimum, a successful replay can disrupt operations by presenting parties with messages that appear genuine but are not.

Lists the following examples of replay attacks:

● **Simple replay:** The opponent simply copies a message and replays it later.

● **Repetition that can be logged:** An opponent can replay a timestamped message within the valid time window.

● **Repetition that cannot be detected:** This situation could arise because the original message could have been suppressed and thus did not arrive at its destination; only the replay message arrives.

● **Backward replay without modification:** This is a replay back to the message sender. This attack is possible if symmetric encryption is used and the sender cannot easily recognize the difference between messages sent and messages received on the basis of content.

One approach to coping with replay attacks is to attach a sequence number to each message used in an authentication exchange. A new message is accepted only if its sequence number is in the proper order.

The difficulty with this approach is that it requires each party to keep track of the last sequence number for each claimant it has dealt with. Because of this overhead, sequence numbers are generally not used for authentication and key exchange. Instead, one of the following two general approaches is used:

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

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

It can be argued (e.g., [LAM92a]) that the timestamp approach should not be used for connection oriented applications because of the inherent difficulties with this technique.

First, some sort of protocol is needed to maintain synchronization among the various processor clocks. This protocol must be both fault tolerant, to cope with network errors, and secure, to cope with hostile attacks.

Second, the opportunity for a successful attack will arise if there is a temporary loss of synchronization resulting from a fault in the clock mechanism of one of the parties.

Finally, because of the variable and unpredictable nature of network delays, distributed clocks cannot be expected to maintain precise synchronization. Therefore, any timestamp-based procedure must allow for a window of time sufficiently large to accommodate network delays yet sufficiently small to minimize the opportunity for attack.

On the other hand, the challenge-response approach is unsuitable for a connectionless type of application because it requires the overhead of a handshake before any connectionless transmission, effectively negating the chief characteristic of a connectionless transaction. For such applications, reliance on some sort of secure time server and a consistent attempt by each party to keep its clocks in synchronization may be the best approach (e.g., [LAM92b]).

Symmetric Encryption Approaches

As was discussed in Section 7.3, a two-level hierarchy of symmetric encryption keys can be used to provide confidentiality for communication in a distributed environment. In general, this strategy involves the use of a trusted key distribution center (KDC). Each party in the network shares a secret key, known as a master key, with the KDC. The KDC is responsible for generating keys to be used for a short time over a connection between two parties, known as session keys, and for distributing those keys using the master keys to protect the distribution. This approach is quite common. As an example, we look at the Kerberos system in Chapter 14. The discussion in this subsection is relevant to an understanding of the Kerberos mechanisms.

Public-Key Encryption Approaches

one approach to the use of public-key encryption for the purpose of session key distribution. This protocol assumes that each of the two parties is in possession of the current public key of the other. It may not be practical to require this assumption

In this case, the central system is referred to as an authentication server (AS), because it is not actually responsible for secret key distribution. Rather, the AS provides public-key certificates. The session key is chosen and encrypted by A; hence, there is no risk of exposure by the AS. The timestamps protect against replays of compromised keys.

One-Way Authentication

One application for which encryption is growing in popularity is electronic mail (e-mail). The very nature of electronic mail, and its chief benefit, is that it is not necessary for the sender and receiver to be online at the same time. Instead, the e-mail message is forwarded to the receiver's electronic mailbox, where it is buffered until the receiver is available to read it. The "envelope" or header of the e-mail message must be in the clear, so that the message can be handled by the store-and-forward e-mail protocol, such as the Simple Mail Transfer Protocol (SMTP) or X.400. However, it is often desirable that the mail-handling protocol not require access to the plaintext form of the message, because that would require trusting the mail-handling mechanism. Accordingly, the e-mail message should be encrypted such that the mail-handling system is not in possession of the decryption key. A second requirement is that of authentication. Typically, the recipient wants some assurance that the message is from the alleged sender.

Symmetric Encryption Approach

Using symmetric encryption, the decentralized key distribution scenario illustrated in Figure 7.11 is impractical. This scheme requires the sender to issue a request to the intended recipient, await a response that includes a session key, and only then send the message. With some refinement, the KDC strategy illustrated in Figure 7.9 is a candidate for encrypted electronic mail. Because we wish to avoid requiring that the recipient (B) be on line at the same time as the sender (A), steps 4 and 5 must be eliminated. For a message with content *M*, the sequence is as follows:

1. A KDC: $IDA||IDB||N1$

2. KDC A: $E(Ka, [Ks||IDB||N1||E(Kb, [Ks||IDA])])$

3. A B: $E(Kb, [Ks||IDA])||E(Ks, M)$

This approach guarantees that only the intended recipient of a message will be able to read it. It also provides a level of authentication that the sender is A. As specified, the protocol does not protect against replays. Some measure of defense could be provided by including a timestamp with the message. However, because of the potential delays in the e-mail process, such timestamps may have limited usefulness.

POSSIBLE QUESTIONS

PART A (20 x 1 = 20 marks)

(ONLINE EXAMINATION)

PART B (5 x 6 = 30 marks)

1. Write a brief note on Authentication function and its various classes?
2. Describe in detail Digital Signature and its properties
3. Mention about the basic uses of message authentication code.
4. i) What are the properties a hash function must satisfy?

   ii) What is the use of authentication protocols?
5. Explain about MAC with suitable diagram.
6. Write a brief note on requirements for a hash function and its properties.
7. Explain in detail Security of Hash Functions and MAC's.

PART – C (1 x 10 = 10 marks)

1. Explain about MAC with suitable diagram
2. Explain about Digital Signatures
3. Describe about Hash function and its security

KARPAGAM ACADEMY OF HIGHER EDUCATION

DEPARTMENT OF COMPUTER SCIENCE

I M.Sc CS

CRYPTOGRAPHY AND NETWORK SECURITY

UNIT 4

S.NO	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	_____ modification to a sequence of messages	timing	content	sequence	none	sequence
2	_____ is a mechanism or service used to verify the integrity of a message	message transformation	message authentication	sequence	traffic	message authentication
3	_____ encryption provides authentication among those who share the secret key.	symmetric	asymmetric	traffic	none	symmetric
4	A_____ is an algorithm that requires the use of a secret key	MAC	MCA	MAB	MAD	MAC
5	A____ function maps a variable-length message into a fixed length value	symmetric	asymmetric	hash	none	hash
6	_____ the ciphertext of the entire message and a serves as its authenticator	message authentication	message transformation	sequence	none	message authentication
7	the straight forward use of _____ key encryption provides confidentiality but not authentication	private	public	protected	none	public
8	An alternative authentication technique involves the use of a secret key to generate a small fixed size block of	encryption	Cryptographic checksum	decryption	traffic	Cryptographic checksum
9	the hash code is also referred to as a _____	message code	authentication	message digest	sequence	message digest
10	the _____is a function of all the bits of the message and provides an error capability	process	creation	detection	insertion	detection
11	A MAC is generated by a function C of the form	MAC=C(K,M)	MAC=C(K,N)	MAC=E(K,M)	MAC=D(K,N)	MAC=C(K,M)
12	the data authentication algorithm , based on _____has been one of the most widely used MACs	AES	DES	DFS	AEM	DES
13	A hash value h is generated by a function H of the form	h=H(M)	h=H(N)	h=H(NM)	h=H(MM)	h=H(M)
14	the purpose of a hash function is to produce a_____ of a file message or other block of data.	HASH	fingerprint	message digest	none	fingerprint
15	for any given value h it is computionally infeasible to find x such that H(x)=h. This is sometimes referred to the	one-way property	two way property	multi way property	none	one-way property
16	for any given block x,it is computationally infeasible to find 'y not equal x' with H(y)=H(x) referred to	one-way property	weak collision resistance	strong collision resistance	none	weak collision resistance
17	it is computationally infeasible to find any pair(x,y)such that H(x)=H(y) is referred to as	strong collision resistance	weak collision resistance	one-way property	none	strong collision resistance
18	_____protects two parties who exchange messages from any third party	message transformation	message authentication	sequence	none	message authentication
19	The ____ must be a bit pattern that depends on the message being signed	digital signature	message authentication	sequence	none	digital signature
20	_____ may be formed by encrypting the entire message with the sender private key	message authentication	digital signature	sequence	none	digital signature
21	_____ is an NIST standard that uses the secure hash algorithms	DSS	AES	DES	RSA	DSS
22	In _____ authentication the recipient wants some assurance that a message is from the alleged sender	Two way	multi way	one way	three way	one way
23	_____,the opponent simply copies a message and replays it later	repetition that can be logged	simple reply	timing reply	none	simple reply
24	_____,the opponent can replay a timestamped message within the valid time window	repetition that can be logged	simple reply	timing reply	none	repetition that can be logged
25	A digital signature is _____.	scanned signature	signature in binary form	encrypting information	handwritten signature	encrypting information
26	_____ is a popular session key creator protocol that requires an authentication server	KDC	Kerberos	CA	none of the above	Kerberos
27	A digital signature needs a(n) _____ system.	symmetric key	asymmetric key	either a or b	neither a or b	asymmetric key
28	Digital signature cannot provide ____ for the message.	integrity	confidentiality	nonrepudiation	authentication	confidentiality
29	Digital signature provides _____.	authentication	nonrepudiation	authentication and	neither a nor b	authentication and nonrepudiation
30	A _____ signature is included in the document; a _____ signature is a separate entity.	conventional; digital	digital; digital	either a or b	either a or b	digital; digital
31	A hash function must meet _____ criteria.	TWO	THREE	FOUR	none of the above	THREE
32	A(n) _____ can be used to preserve the integrity of a document or a message.	message digest	message summary	message confidentiality	none of the above	message digest
33	____ means to prove the identity of the entity that tries to access the system's resources.	message authentication	entity authentication	message confidentiality	none of the above	entity authentication
34	_____ means that a sender must not be able to deny sending a message that he sent.	Confidentiality	integrity	authentication	none of the above	none of the above
35	Message ___ means that the receiver is ensured that the message is coming from the intended sender, not an	Confidentiality	integrity	authentication	none of the above	authentication
36	Message ____ means that the data must arrive at the receiver exactly as sent.	Confidentiality	integrity	authentication	none of the above	integrity
37	Message ____ means that the sender and the receiver expect privacy.	Confidentiality	integrity	authentication	none of the above	Confidentiality
38	MACs are also called	testword	checkword	testbits	checksum	checksum
39	MAC is a	one-to-one mapping	many-to-one mapping	onto mapping	none of the mentioned	many-to-one mapping
40	SHF stands for	symmetric hash function	simple hash funct	secure hash fun	socket hash functio	symmetric hash function
41	_____ cannot be used for encryption or key exchange	RSAA	AES	DES	RSA	RSAA
42	What is data encryption standard (DES)?	block cipher	stream cipher	both	none	block cipher

43	Which one of the following is a cryptographic protocol used to secure HTTP connection?	resource reservation protocol	transport layer security (TSL)	both	none	transport layer security (TSL)
44	Cryptographic hash function takes an arbitrary block of data and returns	fixed size bit string	variable size bit string	both	none	fixed size bit string
45	DES key size	56-bit	168-bit	128-bit	196-bit	56-bit
46	Triple DES key size	56-bit	168-bit	128-bit	196-bit	168-bit
47	AES key size	56-bit	168-bit	128-bit	196-bit	128-bit
48	One commonly used public-key cryptography method is the _____ algorithm.	RSA	DES	RAS	AES	RSA
49	Secure hash algorithm developed by_____	National Institute of standards and	Nation Institute of standards and	National Institute of	National Indian of standards and	National Institute of standards and
50	The most commonly used conventional encryption algorithms is	Blog ciphers	Block ciphers	stream ciphers	none	Block ciphers
51	_____ release of message contents to any person	disend	traffic	disclosure	none	disclosure
52	_____ discovery of the pattern of traffic between parties	traffic analysis	disclosure	timing	none	traffic analysis
53	_____ insertion of messages into the network from a fraudulent source	masquerade	timing	traffic	sequence	masquerade
54	_____modification changes to the contents of a message	traffic analysis	sequence	timing	content	content
55	The _____ method provides a one-time session key for two parties.	Diffie-Hellman	DES	RAS	AES	Diffie-Hellman
56	A(n) _____ is a keyless substitution cipher with N inputs and M outputs that uses a formula to define the	S-box	y-box	Z-BOX	None	S-box
57	_____ is a round cipher based on the Rijndael algorithm that uses a 128-bit block of data.	Diffie-Hellman	DES	RAS	AES	AES
58	_____ is an authentication service developed as part of project athena at MIT	secure	X.509	kerberos	none	kerberos
59	The Caesar cipher is a _____cipher that has a key of 3.	transposition	additive	shift	none	shift
60	In an asymmetric-key cipher, the receiver uses the _____ key.	private	public	Both	none	Private

UNIT –V

Syllabus

Network Security Applications - Authentication Applications – KERBEROS – X.509 Authentication Service – Public Key Infrastructure – Electronic Mail Security – Pretty Good Privacy – S/MIME – IP Security.

NETWORK SECURITY

Kerberos is an authentication service designed for use in a distributed environment.

● Kerberos makes use of a trusted third-part authentication service that enables clients and servers to establish authenticated communication.

AUNTENTICATION APPLICATIONS : 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.

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 a 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 that proposed by 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, then the authentication service is secure if the Kerberos server itself is secure.

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 places a

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: IDc||Pc||IDv
2. AS >> C: Ticket
3. C >> V: IDc||Ticket

Ticket= EKv (IDc||ADc||IDv)

AS	: Authentication Server	V	: Server
IDc	: ID of the client	IDv	: ID of the server
Kv	: secret key shared by AS and V	Pc	:Password of the client
ADc	: Address of client	||	: concatenation

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, and anew server, known as ticket granting server (TGS). The hypothetical scenario is as follows:

Once per user logon session:

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

Once per type of service:

3. C >> TGS: IDc||IDv||$Ticket_{tgs}$
4. TGS >> C: $ticket_v$

Once per service session:

5. C >> V: IDc||$ticket_v$

$Ticket_{tgs}$= Ek_{tgs}(IDc||ADc||IDtgs||TS1||Lifetime1)

$Ticket_V$= Ek_V(IDc||ADc||IDv||TS2||Lifetime2)

C	: Client	V	: Server
IDc	: ID of the client	AS	: Authentication Server
Pc	:Password of the client	ADc	: Address of client
Kv	: secret key shared by AS and V	IDv	: ID of the server
||	: 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.

Kerbero 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 granting ticket TGT)
  - have a Ticket Granting server (TGS)
    - 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 → AS: $ID_c \| ID_{tgs} \| TS_1$

(2) AS → C: $E_{K_c}[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}]$

$$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2]$$

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

(3) C → TGS: $ID_v \| Ticket_{tgs} \| Authenticator_c$

(4) TGS → C: $E_{K_{c,tgs}}[K_{c,v} \| ID_v \| TS_4 \| Ticket_v]$

$$Ticket_{tgs} = E_{K_{tgs}}[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2]$$

$$Ticket_v = E_{K_v}[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4]$$

$$Authenticator_c = E_{K_{tgs}}[ID_C \| AD_C \| TS_3]$$

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

(5) C → V: $Ticket_v \| Authenticator_c$

(6) V → C: $E_{K_{c,v}}[TS_5 + 1]$ (for mutual authentication)

$$Ticket_v = E_{K_v}[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4]$$

$$Authenticator_c = E_{K_{c,v}}[ID_C \| AD_C \| TS_5]$$

Table 4.1 illustrates the mode of dialogue in V4

Message (1)	**Client requests ticket-granting ticket**
ID_C	Tells AS identity of user from this client
ID_{tgs}	Tells AS that user requests access to TGS
TS_1	Allows AS to verify that client's clock is synchronized with that of A
Message (2)	**AS returns ticket-granting ticket**
K_c	Encryption is based on user's password, enabling AS and client to verify password, and protecting contents of message (2)
$K_{c,tgs}$	Copy of session key accessible to client created by AS to permit secu exchange between client and TGS without requiring them to share a permanent key
ID_{tgs}	Confirms that this ticket is for the TGS
$K_{c,tgs}$	
ID_{tgs}	Confirms that this ticket is for the TGS

TS_2	Informs client of time this ticket was issued
$Lifetime_2$	Informs client of the lifetime of this ticket
$Ticket_{tgs}$	Ticket to be used by client to access TGS
	(a) Authentication Service Exchange
Message (3)	**Client requests service-granting ticket**
ID_V	Tells TGS that user requests access to server V
$Ticket_{tgs}$	Assures TGS that this user has been authenticated by AS
$Authenticator_c$	Generated by client to validate ticket
Message (4)	**TGS returns service-granting ticket**
$K_{c,tgs}$	Key shared only by C and TGS protects contents of message (4)
$K_{c,v}$	Copy of session key accessible to client created by TGS to permit secure exchange between client and server without requiring them to share a permanent key
ID_V	Confirms that this ticket is for server V
TS_4	Informs client of time this ticket was issued
$Ticket_V$	Ticket to be used by client to access server V
$Ticket_{tgs}$	Reusable so that user does not have to reenter password
K_{tgs}	Ticket is encrypted with key known only to AS and TGS, to prevent tampering
$K_{c,tgs}$	Copy of session key accessible to TGS used to decrypt authenticator thereby authenticating ticket
ID_C	Indicates the rightful owner of this ticket

AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket
ID_{tgs}	Assures server that it has decrypted ticket properly
TS_2	Informs TGS of time this ticket was issued
$Lifetime_2$	Prevents replay after ticket has expired
$Authenticator_c$	Assures TGS that the ticket presenter is the same as the client for whom the ticket was issued has very short lifetime to prevent replay
$K_{c,tgs}$	Authenticator is encrypted with key known only to client and TGS, t prevent tamperig
ID_c	Must match ID in ticket to authenticate ticket
AD_c	Must match address in ticket to authenticate ticket
TS_3	Informs TGS of time this authenticator was generated
	(b) Ticket-Granting Service Exchange
Message (5)	**Client requests service**
$Ticket_v$	Assures server that this user has been authenticated by AS
$Authenticator_c$	Generated by client to validate ticket
Message (6)	Optional authentication of server to client
$K_{c,v}$	Assures C that this message is from V
$TS_5 + 1$	Assures C that this is not a replay of an old reply
$Ticket_v$	Reusable so that client does not need to request a new ticket from T for each access to the same server

K_V	Ticket is encrypted with key known only to TGS and server, to preve tampering
$K_{c,v}$	Copy of session key accessible to client; used to decrypt authenticato thereby authenticating ticket
ID_C	Indicates the rightful owner of this ticket
AD_c	Prevents use of ticket from workstation other than one that initially requested the ticket
ID_V	Assures server that it has decrypted ticket properly
TS_4	Informs server of time this ticket was issued
$Lifetime_4$	Prevents replay after ticket has expired
$Authenticator_c$	Assures server that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay
$K_{c,v}$	Authenticator is encrypted with key known only to client and server, prevent tampering
ID_C	Must match ID in ticket to authenticate ticket
AD_c	Must match address in ticket to authenticate ticket
TS_5	Informs server of time this authenticator was generated
	(c) Client/Server Authentication

Kerberos 4 Overview

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:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.
2. 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 realm**.

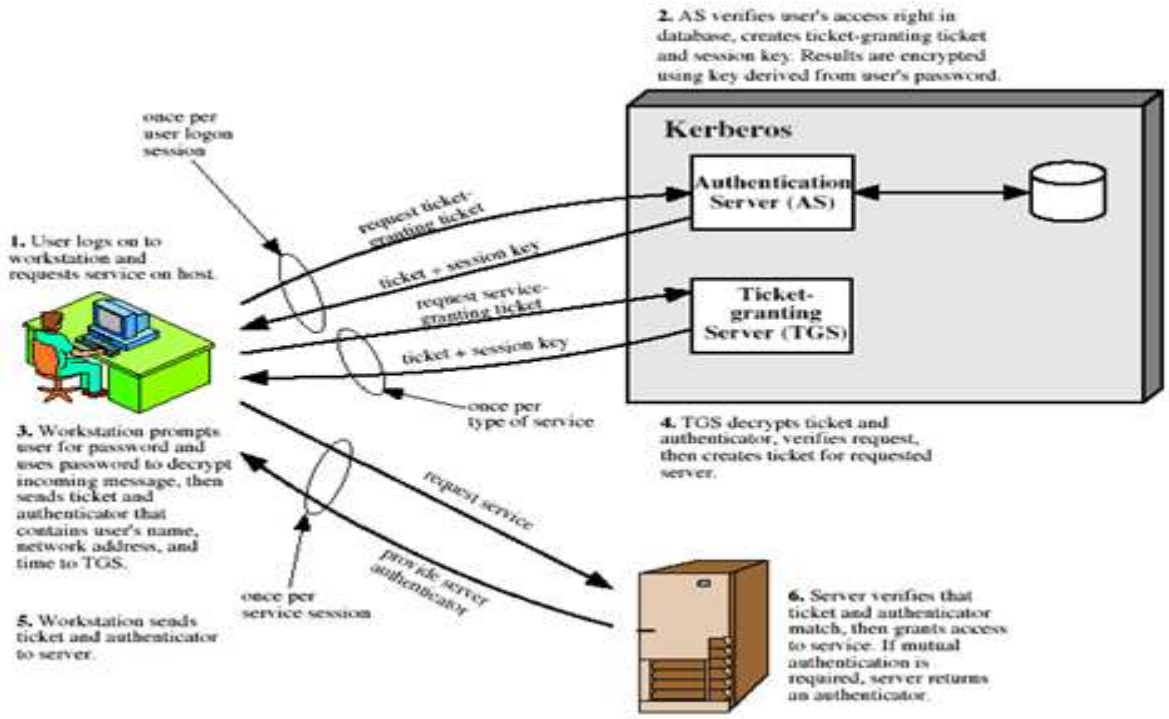


Figure 5.1 Kerberos

The concept of *realm* can be explained as follows.

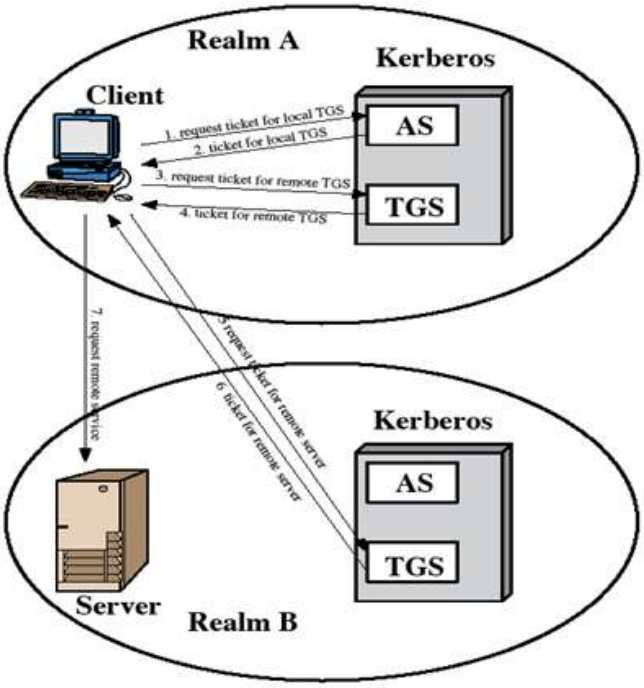


Figure 5.2 Request for Service in Another Realm

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 kept in 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 w ith 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 interrealm authentication. For two realms to support interrealm authentication, a third requirement is added:

3. 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.

Kerberos version 5

Version 5 of Kerberos provides a number of improvements over version 4. • developed in mid 1990's

- provides improvements over v4
  - –addresses environmental shortcomings
  - –and technical deficiencies
- specified as Internet standard RFC 1510

Differences between version 4 and 5

Version 5 is intended to address the limitations of version 4 in two areas:

- **Environmental shortcomings**
  - encryption system dependence
  - internet protocol dependence
  - message byte ordering
  - ticket lifetime
  - authentication forwarding
  - inter-realm authentication
- **Technical deficiencies**
  - double encryption
  - PCBC encryption
  - Session keys
  - Password attacks

The version 5 authentication dialogue

(a) Authentication Service Exchange: to obtain ticket-granting ticket
(1) C → AS: Options ‖ ID_c ‖ $Realm_c$ ‖ ID_{tgs} ‖ $Times$ ‖ $Nonce_1$
(2) AS → C: $Realm_c \| ID_C \| Ticket_{tgs} \| E_{K_c}[K_{c,tgs} \| Times \| Nonce_1 \| Realm_{tgs} \| ID_{tgs}]$
$Ticket_{tgs} = E_{K_{tgs}}[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times]$
(b) Ticket-Granting Service Exchange: to obtain service-granting ticket
(3) C → TGS: Options ‖ ID_v ‖ $Times$ ‖ ‖ $Nonce_2$ ‖ $Ticket_{tgs}$ ‖ $Authenticator_c$
(4) TGS → C: $Realm_c \| ID_C \| Ticket_v \| E_{K_{c,tgs}}[K_{c,v} \| Times \| Nonce_2 \| Realm_v \| ID_V]$
$Ticket_{tgs} = E_{K_{tgs}}[Flags \| K_{c,tgs} \| Realm_c \| ID_C \| AD_C \| Times]$
$Ticket_v = E_{K_v}[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times]$
$Authenticator_c = E_{K_{c,tgs}}[ID_C \| Realm_c \| TS_1]$
(c) Client/Server Authentication Exchange: to obtain service
(5) C → V: Options ‖ $Ticket_v$ ‖ $Authenticator_c$
(6) V → C: $E_{K_{c,V}}$ [TS_2 ‖ Subkey ‖ Seq#]
$Ticket_v = E_{K_v}[Flags \| K_{c,v} \| Realm_c \| ID_C \| AD_C \| Times]$
$Authenticator_c = E_{K_{c,V}}[ID_C \| Realm_c \| TS_2 \| Subkey \| Seq\#]$

First, consider the authentication service exchange. Message (1) is a client request for a ticket-granting ticket. As before, 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
rtime: 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 replayed 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.

These flags introduce significant new functionality to version 5. For now, we defer a discussion of these flags and concentrate on the overall structure of the version 5 protocol.

Let us now compare the ticket-granting service exchange for versions 4 and 5. We see that message (3) for both versions includes 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.

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 ($K_{c,v}$) 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 may be sequence numbered to detect replays.

If mutual authentication is required, the server responds with message (6). This message includes the timestamp from the authenticator. Note that in version 4, the timestamp was incremented by one. This is not necessary in version 5 because the nature of the format of messages is such that it is not possible for an opponent to create message (6) without knowledge of the appropriate encryption keys.

Ticket Flags

The flags field included in tickets in version 5 supports expanded functionality compared to that available in version 4.

X.509 CERTIFICATES OVERVIEW:

- X.509 defines the format for public-key certificates. This format is widely used in a variety of applications.
- A public key infrastructure (PKI) is defined as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.

● Typically, PKI implementations make use of X.509 certificates.

• issued by a Certification Authority (CA), containing:
– version (1, 2, or 3)
– serial number (unique within CA) identifying certificate
–signature algorithm identifier
– issuer X.500 name (CA)

– period of validity (from - to dates)

– subject X.500 name (name of owner)

– subject public-key info (algorithm, parameters, key) – issuer unique identifier (v2+)
– subject unique identifier (v2+) – extension fields (v3)
– signature (of hash of all fields in certificate)

• **notation CA<<A>> denotes certificate for A signed by CA**

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 an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety of contexts. For example, the X.509 certificate format is used in S/MIME), IP Security and SSL/TLS and SET

X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends RSA. The digital signature scheme is assumed to require the use of a hash function.

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.

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 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 the subject in the event the X.500 name has been reused for different entities.

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 identifie

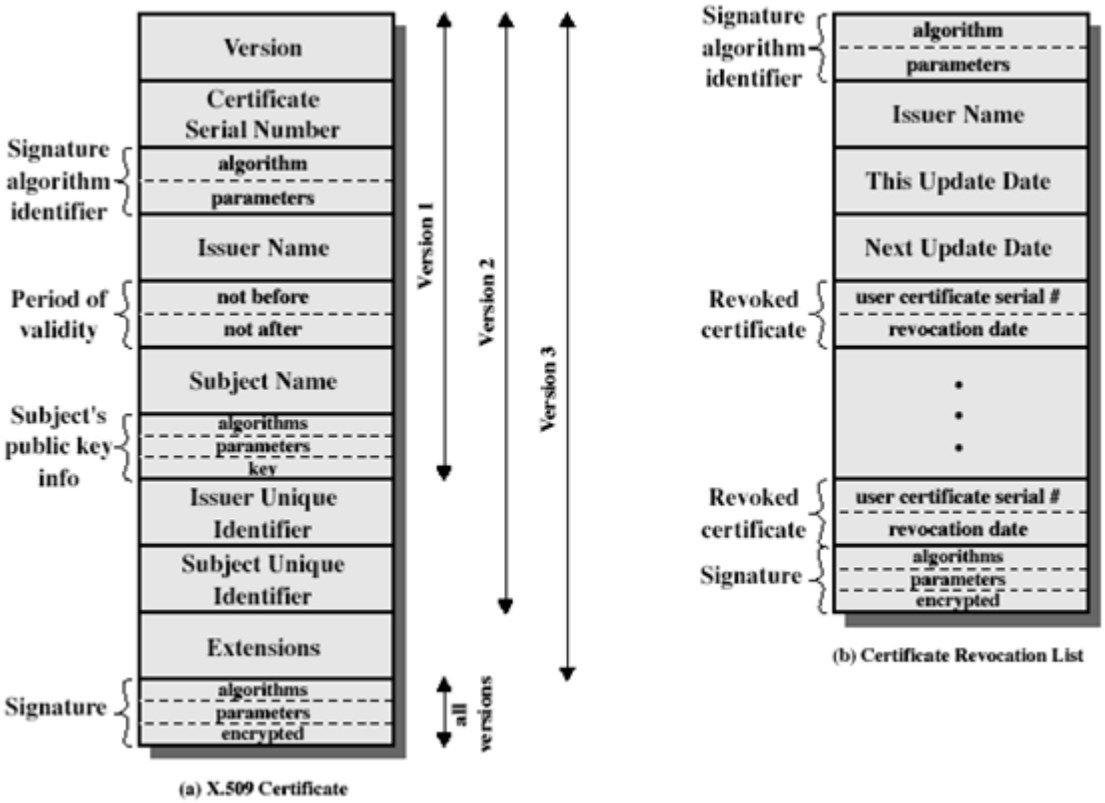


Figure 5.3 : X.509 Formats

The standard uses the following notation to define a certificate:

$$CA\langle\langle A\rangle\rangle = CA \{V, SN, AI, CA, T_A, A, Ap\}$$

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

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

Obtaining a User's 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 without 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.

If there is a large community of users, it may not be practical for all users to subscribe to the same CA. Because it is the CA that signs certificates, each participating user must have a copy of the CA's own public key to verify signatures. This public key must be provided to each user in an absolutely secure (with respect to integrity and authenticity) way so that the user has confidence in the associated certificates. Thus, with many users, it may be more practical for there to be a number of CAs, each of which securely provides its public key to some fraction of the users.

Now suppose that A has obtained a certificate from certification authority X_1 and B has obtained a certificate from CA X_2. If A does not securely know the public key of X_2, then B's certificate, issued by X_2, is useless to A.

A can read B's certificate, but A cannot verify the signature. However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key:

A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as

$$X_1\langle\langle X_2\rangle\rangle\ X_2\ \langle\langle B\rangle\rangle$$

In the same fashion, B can obtain A's public key with the reverse chain:

$$X_2\langle\langle X_1\rangle\rangle\ X_1\ \langle\langle A\rangle\rangle$$

This scheme need not be limited to a chain of two certificates. An arbitrarily long path of CAs can be followed to produce a chain. A chain with N elements would be expressed as

$$X_1\langle\langle X_2\rangle\rangle\ X_2\ \langle\langle X_3\rangle\rangle \ldots\ X_N\langle\langle B\rangle\rangle$$

In this case, each pair of CAs in the chain (X_i, X_{i+1}) must have created certificates for each other.

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.

Figure 4.5, taken from X.509, is an example of such a hierarchy. 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

CA HIERARCHY USE

In the example given below , user A can acquire the following certificates from the directory to establish a certification path to B:

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

When A has obtained these certificates, it can unwrap the certification path in sequence to recover a trusted copy of B's public key. Using this public key, A can send encrypted messages to B. If A wishes to receive encrypted messages back from B, or to sign messages sent to B, then B will require A's public key, which can be obtained from the following certification path:

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

B can obtain this set of certificates from the directory, or A can provide them as part of its initial message to B.

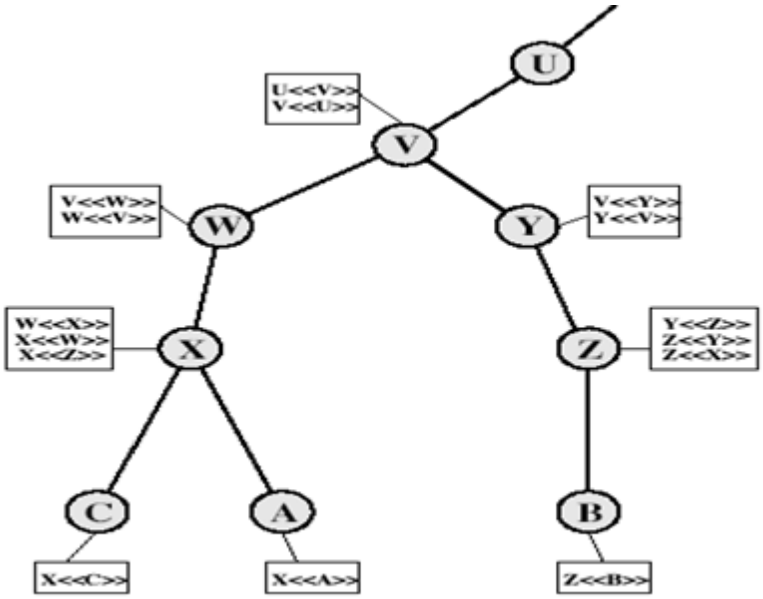


Figure 5.4 X.509 Hierarchy: A Hypothetical Example

CERTIFICATE REVOCATION

- certificates have a period of validity
- may need to revoke before expiry, for the following reasons

eg: 1. user's private key is compromised
2. user is no longer certified by this CA
3. CA's certificate is compromised

- CA's maintain list of revoked certificates

1. the Certificate Revocation List (CRL)

- users should check certs with CA's CRL

AUTHENTICATION PROCEDURES

X.509 includes three alternative authentication procedures:

- **One-Way Authentication**
- **Two-Way Authentication**

- **Three-Way Authentication**

All use public-key signatures

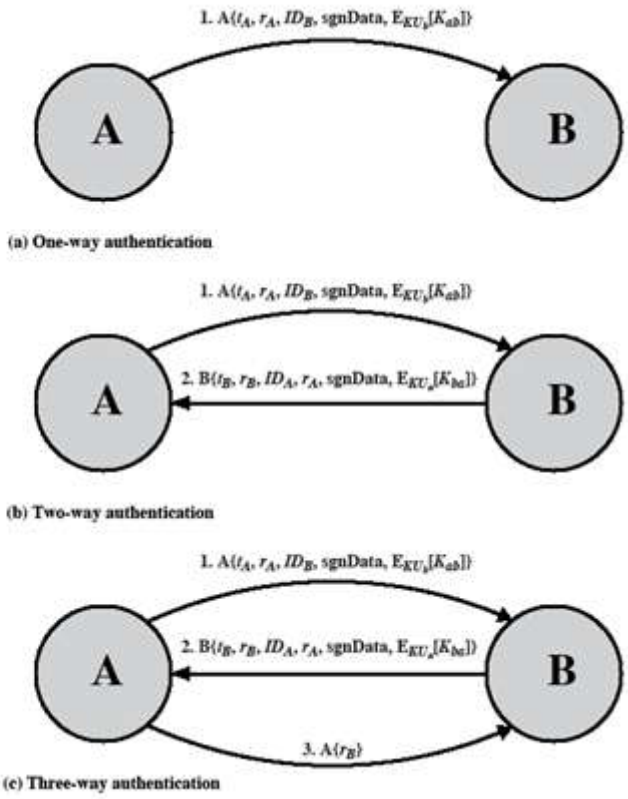


Figure 5.5: X.509 Strong Authentication Procedures

One-Way Authentication

•1 message (A->B) used to establish

– the identity of A and that message is from A
– message was intended for B
– integrity & originality of message

•message must include timestamp, nonce, B's identity and is signed by A

Two-Way Authentication

• 2 messages (A->B, B->A) which also establishes in addition:
– the identity of B and that reply is from B
– that reply is intended for A

– integrity & originality of reply

• reply includes original nonce from A, also timestamp and nonce from B

Three-Way Authentication

• 3 messages (A->B, B->A, A->B) which enables above authentication without

synchronized clocks

- has reply from A back to B containing signed copy of nonce from B
- means that timestamps need not be checked or relied upon

X.509 VERSION 3

` The X.509 version 2 format does not convey all of the information that recent design and implementation experience has shown to be needed. T h e f o llowing requirements not satisfied by version 2:

1. The Subject field is inadequate to convey the identity of a key owner to a public-key user.
2. The Subject field is also inadequate for many applications, which typically recognize entities by an Internet e-mail address, a URL, or some other Internet-related identification.
3. There is a need to indicate security policy information. There is a need to limit the damage that can result from a faulty or malicious CA by setting constraints on the applicability of a particular certificate.
4. It is important to be able to identify different keys used by the same owner at different times.

The certificate extensions fall into three main categories: key and policy information, subject and issuer attributes, and certification path constraints.

Key and Policy Information

These extensions convey additional information about the subject and issuer keys, plus indicators of certificate policy.. For example, a policy might be applicable to the authentication of electronic data interchange (EDI) transactions for the trading of goods within a given price range.

This area includes the following:

 Authority key identifier: Identifies the public key to be used to verify the signature on this certificate or CRL.
 Subject key identifier: Identifies the public key being certified. Useful for subject key pair updating.
 Key usage: Indicates a restriction imposed as to the purposes for which, and the policies under which, the certified public key may be used.
 Private-key usage period: Indicates the period of use of the private key corresponding to the public key.. For example, with digital signature keys, the usage period for the signing private key is typically shorter than that for the verifying public key.
 Certificate policies: Certificates may be used in environments where multiple policies apply.

 Policy mappings: Used only in certificates for CAs issued by other CAs.

Certificate Subject and Issuer Attributes

These extensions support alternative names, in alternative formats, for a certificate subject or certificate issuer and can convey additional information about the certificate subject, to increase a certificate user's confidence that the certificate subject is a particular person or entity. For example, information such as postal address, position within a corporation, or picture image may be required.

The extension fields in this area include the following:

 Subject alternative name: Contains one or more alternative names, using any of a variety of forms

 Subject directory attributes: Conveys any desired X.500 directory attribute values for the subject of this certificate.

Certification Path Constraints

These extensions allow constraint specifications to be included in certificates issued for CAs by other CAs.

The extension fields in this area include the following:

 Basic constraints: Indicates if the subject may act as a CA. If so, a certification path length constraint may be specified.

 Name constraints: Indicates a name space within which all subject names in subsequent certificates in a certification path must be located.

 Policy constraints: Specifies constraints that may require explicit certificate policy identification or inhibit policy mapping for the remainder of the certification path.

ELECTRONIC MAIL SECURITY

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; e-mail compatibility using the radix-64 encoding scheme; and segmentation and reassembly to accommodate long e-mails.

- PGP incorporates tools for developing a public-key trust model and public-key certificate management.
- S/MIME is an Internet standard approach to e-mail security that incorporates the same functionality as PGP.

PRETTY GOOD PRIVACY (PGP)

PGP provides the confidentiality and authentication service that can be used for electronic mail and file storage applications. The steps involved in PGP are

 Select the best available cryptographic algorithms as building blocks.

 Integrate these algorithms into a general purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands.

 Make the package and its documentation, including the source code, freely available via the internet, bulletin boards and commercial networks.

 Enter into an agreement with a company to provide a fully compatible, low cost commercial version of PGP.

PGP has grown explosively and is now widely used. A number of reasons can be cited for this growth.

 It is available free worldwide in versions that run on a variety of platform.

 It is based on algorithms that have survived extensive public review and are considered extremely secure.

e.g., RSA, DSS and Diffie Hellman for public key encryption CAST-128, IDEA and 3DES for conventional encryption SHA-1 for hash coding.

 It has a wide range of applicability.

 It was not developed by, nor it is controlled by, any governmental or standards organization.

Operational description

The actual operation of PGP consists of five services: authentication, confidentiality, compression, e-mail compatibility and segmentation.

1. Authentication

The sequence for authentication is as follows: The sender creates the message

 SHA-1 is used to generate a 160-bit hash code of the message

 The hash code is encrypted with RSA using the sender's private key and the result is prepended to the message

 The receiver uses RSA with the sender's public key to decrypt and recover the hash code.

 The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.

2. Confidentiality

Confidentiality is provided by encrypting messages to be transmitted or to be stored locally as files. In both cases, the conventional encryption algorithm CAST-128 may be used. The 64-bit cipher feedback (CFB) mode is used. In PGP, each conventional key is used only once. That is, a new key is generated as a random 128-bit number for each message. Thus although this is referred to as **a session key**, it is in reality a **one time key**. To protect the key, it is encrypted with the receiver's public key.

The sequence for confidentiality is as follows:

 The sender generates a message and a random 128-bit number to be used as a session key for this message only.
 The message is encrypted using CAST-128 with the session key.

 The session key is encrypted with RSA, using the receiver's public key and is prepended to the message.
 The receiver uses RSA with its private key to decrypt and recover the session key.
 The session key is used to decrypt the message.

Confidentiality and authentication

Here both services may be used for the same message. First, a signature is generated for the plaintext message and prepended to the message. Then the plaintext plus the signature is encrypted using CAST-128 and the session key is encrypted using RSA.

3. Compression

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

The signature is generated before compression for two reasons:

 It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be
necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.
 Even if one were willing to generate dynamically a recompressed message fro verification, PGP's compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and as a result, produce different compression forms.

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. The compression algorithm used is ZIP.

4. e-mail compatibility

Many electronic mail systems only permit the use of blocks consisting of ASCII texts. 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 conversion. Each group of three octets of binary data is mapped into four ASCII characters.

e.g., consider the 24-bit (3 octets) raw text sequence 00100011 01011100 10010001, we can express this input in block of 6-bits to produce 4 ASCII characters.

001000 110101 110010 010001

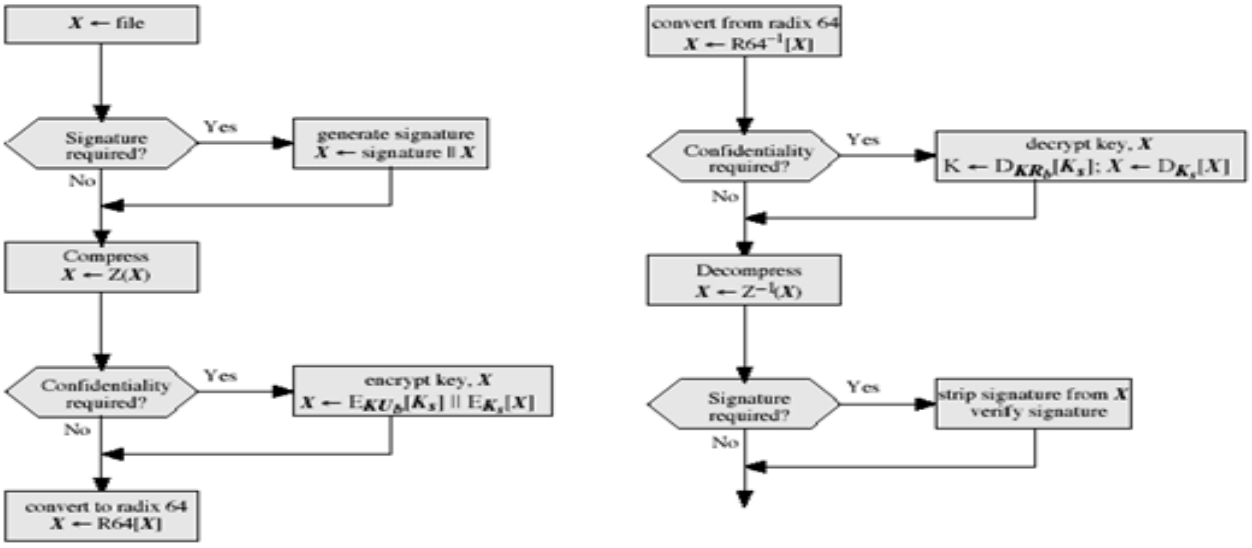
I L Y R => corresponding ASCII characters

5. Segmentation and reassembly

E-mail facilities often are restricted to a maximum length. E.g., many of the facilities accessible through the internet impose a maximum length of 50,000 octets. Any message longer than that must be broken up into smaller segments, each of which is mailed separately.

To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to send via e-mail. The segmentation is done after all the other processing, including the radix-64 conversion. At the receiving end, PGP must strip off all e-mail headers and reassemble the entire original block before performing the other steps.

PGP Operation Summary:



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

Figure 5.6. Transmission and Reception of PGP Messages

Cryptographic keys and key rings

Three separate requirements can be identified with respect to these keys:
 A means of generating unpredictable session keys is needed.
 It must allow a user to have multiple public key/private key pairs.

 Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of correspondents.
We now examine each of the requirements in turn.

1. Session key generation

Each session key is associated with a single message and is used only for the purpose of encryption and decryption of that message. Random 128-bit numbers are generated using CAST-128 itself. The input to the random number generator consists of a 128-bit key and two 64-bit blocks that are treated as plaintext to be encrypted. Using cipher feedback mode, the CAST-128 produces two 64-bit cipher text blocks, which are concatenated to form the 128-bit session key. The plaintext input to CAST-128 is itself derived from a stream of 128-bit randomized numbers. These numbers are based on the keystroke input from the user.

2. Key identifiers

If multiple public/private key pair are used, then how does the recipient know which of the public keys was used to encrypt the session key? One simple solution would be to transmit the public key with the message but, it is unnecessary wasteful of space. Another solution would be to associate an identifier with each public key that is unique at least within each user.

The solution adopted by PGP is to assign a key ID to each public key that is, with very high probability, unique within a user ID. The key ID associated with each public key consists of its least significant 64 bits. i.e., the key ID of public key KU_a is ($KU_a \bmod 2^{64}$).

A message consists of three components.

 Message component – includes actual data to be transmitted, as well as the filename and a timestamp that specifies the time of creation.
 Signature component – includes the following

o Timestamp – time at which the signature was made. o Message digest – hash code.
o Two octets of message digest – to enable the recipient to determine if the correct public key was used to decrypt the message.
o Key ID of sender's public key – identifies the public key

 Session key component – includes session key and the identifier of the recipient public key.

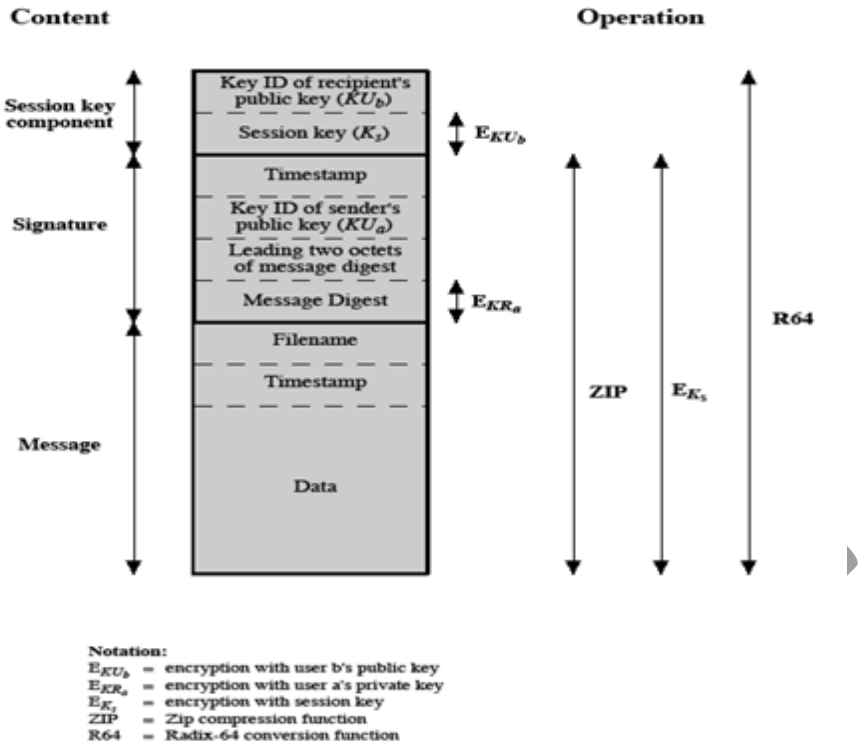


Figure 5.7: General Format of PGP Message (from A to B)

3. Key rings

PGP provides a pair of data structures at each node, one to store the public/private key pair owned by that node and one to store the public keys of the other users known at that node. These data structures are referred to as private key ring and public key ring.

The general structures of the private and public key rings are shown below:

Timestamp – the date/time when this entry was made.

Key ID – the least significant bits of the public key.

Public key – public key portion of the pair.

Private key – private key portion of the pair.

User ID – the owner of the key.

Key legitimacy field – indicates the extent to which PGP will trust that this is a valid public key for this user.

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
• • •	• • •	• • •	• • •	• • •
T_i	PU_i mod 2^{64}	PU_i	E(H(P_i), PR_i)	User i
• • •	• • •	• • •	• • •	• • •

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
T_i	PU_i mod 2^{64}	PU_i	trust_flag$_i$	User i	trust_flag$_i$		
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •

* = field used to index table

Figure 5.8: General Structure of Private and Public Key Rings

Signature trust field – indicates the degree to which this PGP user trusts the signer to certify public key.

Owner trust field – indicates the degree to which this public key is trusted to sign other public key certificates.

PGP message generation

First consider message transmission and assume that the message is to be both signed and encrypted. The sending PGP entity performs the following steps:

1. signing the message

 PGP retrieves the sender's private key from the private key ring using user ID as an index. If user ID was not provided, the first private key from the ring is retrieved.
 PGP prompts the user for the passpharse (password) to recover the unencrypted private key.
 The signature component of the message is constructed.

2. encrypting the message

 PGP generates a session key and encrypts the message.

 PGP retrieves the recipient's public key from the public key ring using user ID as index.
 The session key component of the message is constructed.

The receiving PGP entity performs the following steps:

1. decrypting the message

- PGP retrieves the receiver's private key from the private key ring, using the key ID field in the session key component of the message as an index.
- PGP prompts the user for the passpharse (password) to recover the unencrypted private key.
- PGP then recovers the session key and decrypts the message.

2. Authenticating the message

- PGP retrieves the sender's public key from the public key ring, using the key ID field in the signature key component of the message as an index.
- PGP recovers the transmitted message digest.

- PGP computes the message digest for the received message and compares it to the transmitted message digest to authenticate.

Public-Key Management

This whole business of protecting public keys from tampering is the single most difficult problem in practical public key applications. PGP provides a structure for solving this problem, with several suggested options that may be used.

The Use of Trust

Although PGP does not include any specification for establishing certifying authorities or for establishing trust, it does provide a convenient means of using trust, associating trust with public keys, and exploiting trust information. The basic structure is as follows. Each entry in the public-key ring is a public-key certificate.

We can describe the operation of the trust processing as follows:

1. When A inserts a new public key on the public-key ring, PGP must assign a value to the trust flag that is associated with the owner of this public key. If the owner is A, and therefore this public key also appears in the private-key ring, then a value of ultimate trust is automatically assigned to the trust field. Otherwise, PGP asks A for his assessment of the trust to be assigned to the owner of this key, and A must enter the desired level. The user can specify that this owner is

unknown, untrusted, marginally trusted, or completely trusted.

2. When the new public key is entered, one or more signatures may be attached to it. More signatures may be added later. When a signature is inserted into the entry, PGP searches the public-key ring to see if the author of this signature is among the known public-key owners. If so, the OWNERTRUST value for this owner is assigned to the SIGTRUST field for this signature. If not, an unknown user value is assigned.

3. The value of the key legitimacy field is calculated on the basis of the signature trust fields present in this entry. If at least one signature has a signature trust value of ultimate, then the key legitimacy value is set to complete.

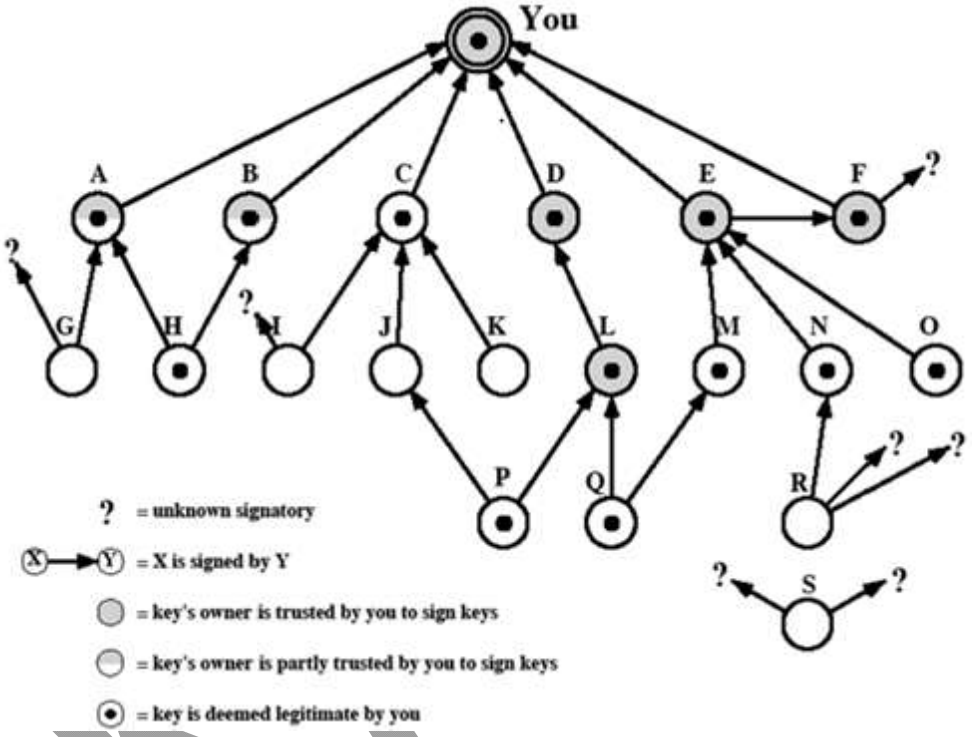


Figure 5.9: PGP Trust Model Example

The node labeled "You" refers to the entry in the public-key ring corresponding to this user. This key is legitimate and the OWNERTRUST value is ultimate trust. Each other node in the key ring has an OWNERTRUST value of undefined unless some other value is assigned by the user. In this example, this user has specified that it always trusts the following users to sign other keys: D, E, F, L. This user partially trusts users A and B to sign other keys.

So the shading, or lack thereof, of the nodes in Figure 4.12 indicates the level of trust assigned by this user. The tree structure indicates which keys have been signed by which other users. If a key is signed by a user whose key is also in this key ring, the arrow joins the signed key to the signatory. If the key is signed by a user whose key is not present in this key ring, the arrow joins the signed key to a question mark, indicating that the signatory is unknown to this

user.

S/MIME

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, based on technology from RSA Data Security. S/MIME is defined in a number of documents, most importantly RFCs 3369, 3370, 3850 and 3851.

Multipurpose Internet Mail Extensions

MIME is an extension to the RFC 822 framework that is intended to address some of the problems and limitations of the use of SMTP (Simple Mail Transfer Protocol) or some other mail transfer protocol and RFC 822 for electronic mail. Following are the limitations of SMTP/822 scheme:

1. SMTP cannot transmit executable files or other binary objects.

2. SMTP cannot transmit text data that includes national language characters because these are represented by 8-bit codes with values of 128 decimal or higher, 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 use a consistent set of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle nontextual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include:
o Deletion, addition, or reordering of carriage return and linefeed o Truncating or wrapping lines longer than 76 characters
o Removal of trailing white space (tab and space characters) o Padding of lines in a message to the same length
o Conversion of tab characters into multiple space characters

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 822 implementations. The specification is provided in RFCs 2045 through 2049.

Overview

The MIME specification includes the following elements:

1. **Five new message header** fields are defined, which may be included in an RFC 822 header. These fields provide information about the body of the message.

2. **A number of content formats** are defined, thus standardizing representations that support multimedia electronic mail.

3. **Transfer encodings** are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

The five header fields defined in MIME are as follows:

 MIME-Version: Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.

 Content-Type: Describes the data contained in the body with sufficient detail

 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.

 Content-ID: Used to identify MIME entities uniquely in multiple contexts.

 Content-Description: A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

MIME Content Types

Table lists the content types specified in RFC 2046. There are seven different major types of content and a total of 15 subtypes

Table. MIME Content Types

Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.

Multipart		together. They should be presented to the receiver in the order th they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for deliverin 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 RF 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-bod	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 kHz.
Application	PostScript	Adobe Postscript.
	octet-stream	General binary data consisting of 8-bit bytes.

MIME Transfer Encodings

The other major component of the MIME specification, in addition to content type specification, is a definition of transfer encodings for message bodies. The objective is to provide reliable delivery across the largest range of environments.

The MIME standard defines two methods of encoding data. The Content-Transfer-Encoding field can actually take on six values, as listed in Table 4.3. For SMTP transfer, it is safe to use the 7bit form. The 8bit and binary forms may be usable in other mail transport contexts. Another Content-Transfer-Encoding value is x-token, which indicates that some other encoding scheme is used, for which a name is to be supplied. The two actual encoding schemes defined are quoted-printable and base64.

Table 4.3 MIME Transfer Encodings

7bit	The data are all represented by short lines of ASCII characters.
8bit	The lines are short, but there may be non-ASCII characters (octets with the high-order bit set).
binary	Not only may non-ASCII characters be present but the lines are not necessaril short enough for SMTP transport.
quoted-printable	Encodes the data in such a way that if the data being encoded are mostly ASC text, the encoded form of the data remains largely recognizable by humans.
base64	Encodes data by mapping 6-bit blocks of input to 8-bit blocks of output, all o which are printable ASCII characters.
x-token	A named nonstandard encoding.

Canonical Form

An important concept in MIME and S/MIME is that of canonical form. Canonical form is a format, appropriate to the content type, that is standardized for use between systems. This is in contrast to native form, which is a format that may be peculiar to a particular system.

S/MIME Functionality

In terms of general functionality, S/MIME is very similar to PGP. Both offer the ability to sign and/or encrypt messages. In this subsection, we briefly summarize S/MIME capability.

Functions

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.

 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.

 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.

Cryptographic Algorithms

• hash functions: SHA-1 & MD5 • digital signatures: DSS & RSA
• session key encryption: ElGamal & RSA • message encryption: Triple-DES, RC2/40 and others
• have a procedure to decide which algorithms to use.

S/MIME Messages

S/MIME makes use of a number of new MIME content types, which are shown in Table 4.5. All of the new application types use the designation PKCS. This refers to a set of public-key cryptography specifications issued by RSA Laboratories and made available for the S/MIME effort.

Table 4.5. S/MIME Content Types

Type	Subtype	smime Parameter	Description
Multipart	Signed		A clear-signed message in two parts: one is the message and the other is the signature.
Application	pkcs 7-mime	signedData	A signed S/MIME entity.
	pkcs 7-mime	envelopedData	An encrypted S/MIME entity.
	pkcs 7-mime	degenerate signedDa	An entity containing only public- key certificates.
	pkcs 7-mime	CompressedData	A compressed S/MIME entity
	pkcs 7-signature	signedData	The content type of the signature subpart of multipart/signed message.

We examine each of these in turn after first looking at the general procedures for S/MIME message preparation.

SECURING A MIME ENTITY

S/MIME secures a MIME entity with a signature, encryption, or both. A MIME entity may be an entire message (except for the RFC 822 headers), or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message. In all cases, the message to be sent is converted to canonical form. In particular, for a given type and subtype, the appropriate canonical form is used for the message content. For a multipart message, the appropriate canonical form is used for each subpart.

The use of transfer encoding requires special attention.

i)EnvelopedData

An application/pkcs7-mime subtype is used for one of four categories of S/MIME processing, each with a unique smime-type parameter. In all cases, the resulting entity, referred to as an object, is represented in a form known as Basic Encoding Rules (BER), which is defined in ITU-T Recommendation X.209. The steps for preparing an envelopedData MIME entity are as follows:

1. Generate a pseudorandom session key for a particular symmetric encryption algorithm (RC2/40 or tripleDES).

2. For each recipient, encrypt the session key with the recipient's public RSA key.

3. For each recipient, prepare a block known as RecipientInfo that contains an identifier of the recipient's public-key certificate,-an identifier of the algorithm used to

encrypt the session key, and the encrypted session key.

This is an X.509 certificate, discussed later in this section.

4. Encrypt the message content with the session key.

The RecipientInfo blocks followed by the encrypted content constitute the envelopedData. This information is then encoded into base64. To recover the encrypted message, the recipient first strips off the base64 encoding. Then the recipient's private key is used to recover the session key. Finally, the message content is decrypted with the session key.

ii)SignedData

The signedData smime-type can actually be used with one or more signers. For clarity, we confine our description to the case of a single digital signature. The steps for preparing a signedData MIME entity are as follows:

1. Select a message digest algorithm (SHA or MD5).
2. Compute the message digest, or hash function, of the content to be signed.
3. Encrypt the message digest with the signer's private key.
4. Prepare a block known as SignerInfo that contains the signer's public-key certificate, an identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest, and the encrypted message digest.

The signedData entity consists of a series of blocks, including a message digest algorithm identifier, the message being signed, and SignerInfo. The signedData entity may also include a set of public-key certificates sufficient to constitute a chain from a recognized root or top-level certification authority to the signer. This information is then encoded into base64.

To recover the signed message and verify the signature, the recipient first strips off the base64 encoding. Then the signer's public key is used to decrypt the message digest. The recipient independently computes the message digest and compares it to the decrypted message digest to verify the signature.

iii)Clear Signing

- Clear signing is achieved using the multipart content type with a signed subtype.
- As was mentioned, this signing process does not involve transforming the message to be signed, so that the message is sent "in the clear."

- Thus, recipients with MIME capability but not S/MIME capability are able to read the incoming message.

A multipart/signed message has two parts. The first part can be any MIME type but must be prepared so that it will not be altered during transfer from source to destination. This means that if the first part is not 7bit, then it needs to be encoded using base64 or quoted-printable. Then this part is processed in the same manner as signedData, but in this case an object with signedData format is created that has an empty message content field. This object is a detached signature. It is then transfer encoded using base64 to become the second part of the multipart/signed message. This second part has a MIME content type of application and a subtype of pkcs7-signature

The protocol parameter indicates that this is a two-part clear-signed entity. The receiver can verify the signature by taking the message digest of the first part and comparing this to the message digest recovered from the signature in the second part.

Registration Request

- Typically, an application or user will apply to a certification authority for a

public-key certificate.

- The application/pkcs10 S/MIME entity is used to transfer a certification request. The certification request includes certificationRequestInfo block, followed by an identifier of the public-key encryption algorithm, followed by the signature of the certificationRequestInfo block, made using the sender's private key.
- The certificationRequestInfo block includes a name of the certificate subject (the entity whose public key is to be certified) and a bit-string representation of the user's public key.

Certificates-Only Message

A message containing only certificates or a certificate revocation list (CRL) can be sent in response to a registration request. The message is an application/pkcs7-mime type/subtype with an smime-type parameter of degenerate. The steps involved are the same as those for creating a signedData message, except that there is no message content and the signerInfo field is empty.

S/MIME Certificate Processing

S/MIME uses public-key certificates that conform to version 3 of X.509 The key-management scheme used by S/MIME is in some ways a hybrid between a strict X.509 certification hierarchy and PGP's web of trust. As with the PGP model, S/MIME managers and/or users must configure each client with a list of trusted keys and with certificate revocation lists.

****User Agent Role***

An S/MIME user has **several key-management functions** to perform:

- **Key generation:** The user of some related administrative utility (e.g., one associated with LAN management) MUST be capable of generating a key pair from a good source of nondeterministic random input and be protected in a secure fashion. A user agent SHOULD generate RSA key pairs with a length in the range of 768 to 1024 bits and MUST NOT generate a length of less than 512 bits.

- **Registration:** A user's public key must be registered with a certification authority in order to receive an X.509 public-key certificate.

- **Certificate storage and retrieval:** A user requires access to a local list of certificates in order to verify incoming signatures and to encrypt outgoing messages. Such a list could be maintained by the user or by some local administrative entity on behalf of a number of users.

****VeriSign Certificates***

There are several companies that provide certification authority (CA) services. For example, Nortel has designed an enterprise CA solution and can provide S/MIME support within an organization. There are a number of Internet-based CAs, including VeriSign, GTE, and the U.S. Postal Service. Of these, the most widely used is the VeriSign CA service, a brief description of which we now provide.

The information contained in a Digital ID depends on the type of Digital ID and its use. At a minimum, each Digital ID contains

- Owner's public key
- Owner's name or alias
- Expiration date of the Digital ID Serial number of the Digital ID
- Name of the certification authority that issued the Digital ID
- Digital signature of the certification authority that issued the Digital ID

Digital IDs can also contain other user-supplied information, including

- Address
- E-mail address
- Basic registration information (country, zip code, age, and gender)

VeriSign provides three levels, or classes, of security for public-key certificates. A user requests a certificate online at VeriSign's Web site or other participating Web sites. Class 1 and Class 2 requests are processed on line, and in most cases take only a few seconds to approve. Briefly, the following procedures are used:

- For Class 1 Digital IDs, VeriSign confirms the user's e-mail address by sending a PIN and Digital ID pick-up information to the e-mail address provided in the application.

- For Class 2 Digital IDs, VeriSign verifies the information in the application through an automated comparison with a consumer database in addition to performing all of the checking associated with a Class 1 Digital ID. Finally, confirmation is sent to the specified postal address alerting the user that a Digital ID has been issued in his or her name.

- For Class 3 Digital IDs, VeriSign requires a higher level of identity assurance. An individual must prove his or her identity by providing notarized credentials or applying in person.

Class	Identity Checks	Usage
1	name/email check	web browsing/email
2	+ enroll/addr check	email, subs, s/w validate

3	+ ID documents	e-banking/service access

Enhanced Security Services

As of this writing, three enhanced security services have been proposed in an Internet draft.:

- **Signed receipts:** A signed receipt may be requested in a SignedData object. Returning a signed receipt provides proof of delivery to the originator of a message and allows the originator to demonstrate to a third party that the recipient received the message.
- **Security labels**: A security label may be included in the authenticated attributes of a SignedData object. A security label is a set of security information regarding the sensitivity of the content that is protected by S/MIME encapsulation. The labels may be used for access control, by indicating which users are permitte access to an object.
- **Secure mailing lists:** When a user sends a message to multiple recipients, a certain amount of per-recipient processing is required, including the use of each recipient's public key. The user can be relieved of this work by employing the services of an S/MIME Mail List Agent (MLA). An MLA can take a single incoming message, perform the recipient-specific encryption for each recipient, and forward the message. The originator of a message need only send the message to the MLA, with encryption performed using the MLA's public key.

IP Security

IP security (IPSec) is a capability that can be added to either current version of the Internet Protocol (IPv4 or IPv6), by means of additional headers.

● IPSec encompasses three functional areas: authentication, confidentiality, and key management.

● Authentication makes use of the HMAC message authentication code. Authentication can be applied to the entire original IP packet (tunnel mode) or to all of the packet except for the IP header (transport mode).

● Confidentiality is provided by an encryption format known as encapsulating security payload. Both tunnel and transport modes can be accommodated.

● IPSec defines a number of techniques for key management.

To provide security, the IAB (Internet Architecture Board) included authentication and encryption as necessary security features in the next-generation IP, which has been issued as IPv6. Fortunately, these security capabilities were designed to be usable both with the current IPv4 and the future IPv6. This means that vendors can begin offering these features now, and many vendors now do have some IPsec capability in their products. The IPsec specification now exists as a set of Internet standards.

Applications of IP security

IPsec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet.

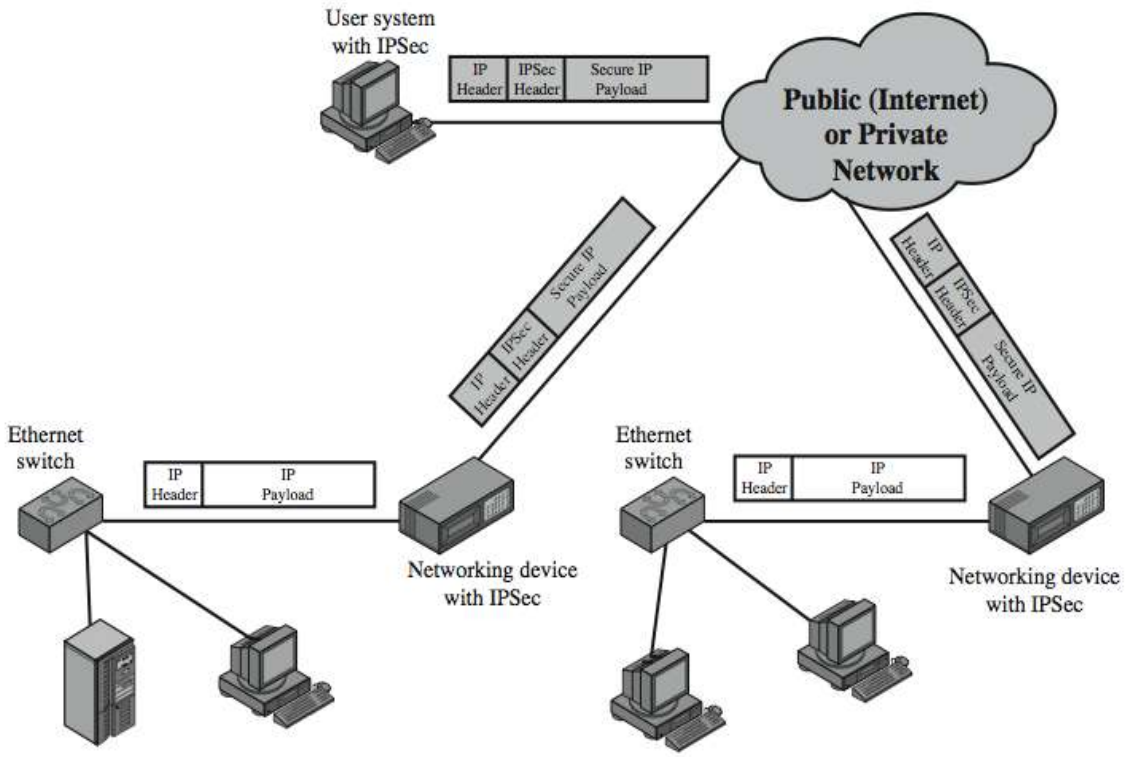
Examples of its use include the following:
- Secure branch office connectivity over the Internet:
- Secure remote access over the Internet:
- Establishing extranet and intranet connectivity with partners:
- Enhancing electronic commerce security

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.

IP Security

- have a range of application specific security mechanisms
  - eg. S/MIME, PGP, Kerberos, SSL/HTTPS
- however there are security concerns that cut across protocol layers
- would like security implemented by the network for all applications
-
- general IP Security mechanisms
- provides
  - authentication
  - confidentiality
  - key management
- applicable to use over LANs, across public & private WANs, & for the Internet
- need identified in 1994 report
  - need authentication, encryption in IPv4 & IPv6
-

IP Security Scenario



➢ Fig 5.10. IP Security Scenario

IP security services

IPsec provides security services at the IP layer by enabling a system to select required security Protocols.

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 designated by the format of the packet for that protocol, Encapsulating Security Payload (ESP). RFC 4301 lists the following services:

1. Access control
2. Connectionless integrity
3. Data origin authentication
4. Rejection of replayed packets (a form of partial sequence integrity)
5. Confidentiality (encryption)
6. Limited traffic flow confidentiality

Benefits of IPSec

- in a firewall/router provides strong security to all traffic crossing the perimeter
- in a firewall/router is resistant to bypass
- is below transport layer, hence transparent to applications
- can be transparent to end users
- can provide security for individual users
- secures routing architecture

IPSec Services

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
  - a form of partial sequence integrity
- Confidentiality (encryption)
- Limited traffic flow confidentiality

IP Security Architecture

- specification is quite complex, with groups:
  - Architecture
    - RFC4301 *Security Architecture for Internet Protocol*
  - Authentication Header (AH)
    - RFC4302 *IP Authentication Header*
  - Encapsulating Security Payload (ESP)
    - RFC4303 *IP Encapsulating Security Payload (ESP)*
  - Internet Key Exchange (IKE)
    - RFC4306 *Internet Key Exchange (IKEv2) Protocol*
  - Cryptographic algorithms
  - Other

Security Associations

A key concept that appears in both the authentication and confidentiality mechanisms for IP is 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.

A security association is uniquely identified by three parameters:

Security Parameters Index (SPI) - The SPI is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed

IP Destination Address - this is the address of the destination endpoint of the SA,

Security Protocol Identifier - This indicates whether the association is an AH or ESP security association.

- SA parameters
  - seq no counter, AH info (Authentication algorithm, keys, key lifetimes) & EH (Encryption and authentication algorithm, keys,), lifetime etc.

Benefits of IPSec

- in a firewall/router provides strong security to all traffic crossing the perimeter
- in a firewall/router is resistant to bypass
- is below transport layer, hence transparent to applications
- can be transparent to end users
- can provide security for individual users
- secures routing architecture
-

IP Security Architecture

- specification is quite complex, with groups:
  - Architecture
    - RFC4301 *Security Architecture for Internet Protocol*
  - Authentication Header (AH)
    - RFC4302 *IP Authentication Header*
  - Encapsulating Security Payload (ESP)
    - RFC4303 *IP Encapsulating Security Payload (ESP)*
  - Internet Key Exchange (IKE)
    - RFC4306 *Internet Key Exchange (IKEv2) Protocol*
  - Cryptographic algorithms
  - Other

Transport and Tunnel Modes

- Transport Mode
  - to encrypt & optionally authenticate IP data
  - can do traffic analysis but is efficient
  - good for ESP host to host traffic
- Tunnel Mode
  - encrypts entire IP packet

- add new header for next hop
- no routers on way can examine inner IP header
- good for VPNs, gateway to gateway security

➢ **Transport and Tunnel Modes**

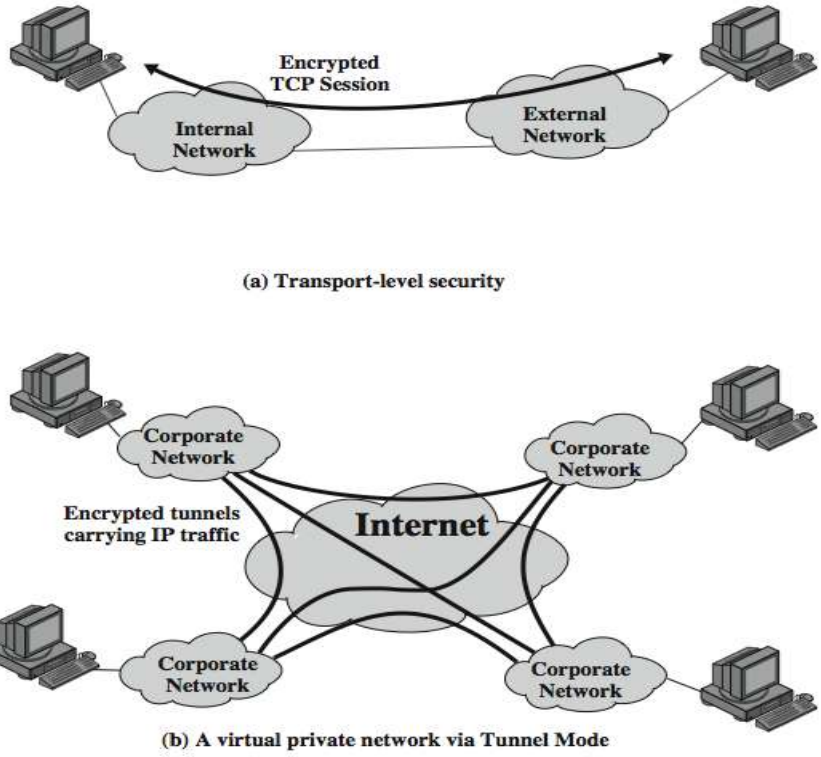


Fig 5.11. Transport and Tunnel Mode Protocols

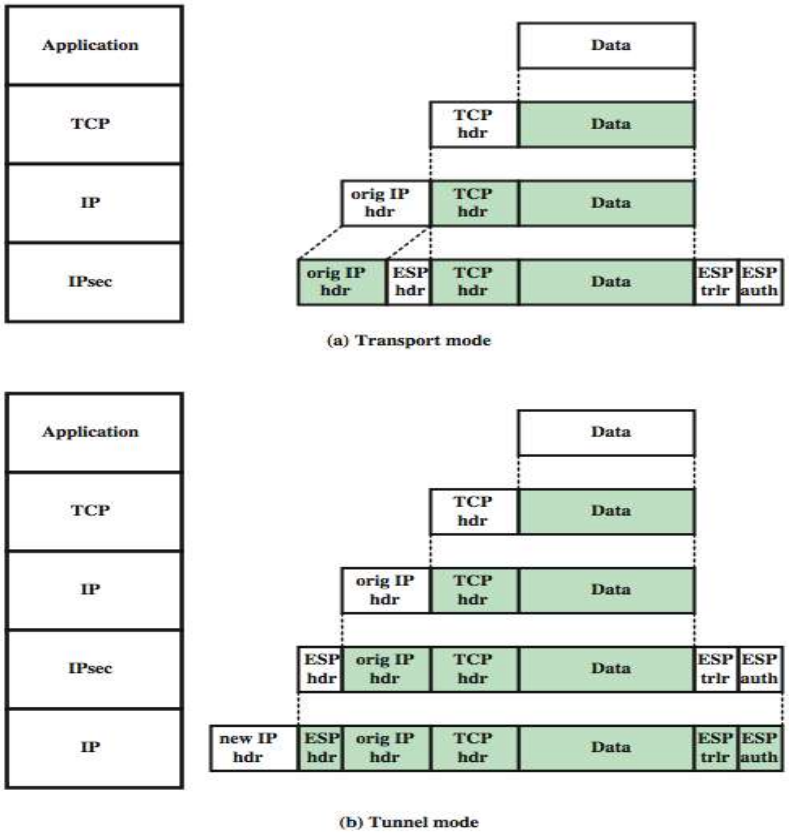


Fig 5.12 Transport and Tunnel mode

Transport Mode

Transport mode provides protection primarily for upper-layer protocols.

That is, transport mode protection extends to the payload of an IP packet.

Tunnel Mode

Tunnel mode provides protection to the entire IP packet.

To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload 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 routers along the way are able to examine the inner IP header.

Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding to the security.

Tunnel mode is used when one or both ends of an SA are a security gateway, such as a firewall or router that implements IPSec.

Key Management

- The key management portion of IPSec involves the determination and distribution of secret keys. A typical requirement is four keys for communication between two applications: transmit and receive pairs
- for both AH and ESP. The IPSec Architecture document mandates support for two types of key management:
- ● **Manual:** A system administrator manually configures each system with its own keys and with the keys of other communicating systems. This is practical for small, relatively static environments.
- ● **Automated:** An automated system enables the on-demand creation of keys for SAs and facilitates the use of keys in a large distributed system with an evolving configuration.
- The default automated key management protocol for IPSec is referred to as ISAKMP/Oakley and consists of the following elements:
- ● **Oakley Key Determination Protocol**: Oakley is a key exchange protocol based on the Diffie-Hellman algorithm but providing added security. Oakley is generic in that it does not dictate specific formats.
- ● **Internet Security Association and Key Management Protocol (ISAKMP):** ISAKMP provides a framework for Internet key management and provides the specific protocol support, including formats, for negotiation of security attributes.

Oakley

- a key exchange protocol
- based on Diffie-Hellman key exchange
- adds features to address weaknesses
  - no info on parties, man-in-middle attack, cost
  - so adds cookies, groups (global params), nonces, DH key exchange with authentication
- can use arithmetic in prime fields or elliptic curve fields

ISAKMP

- Internet Security Association and Key Management Protocol
- provides framework for key management
- defines procedures and packet formats to establish, negotiate, modify, & delete SAs
- independent of key exchange protocol, encryption alg, & authentication method
- IKEv2 no longer uses Oakley & ISAKMP terms, but basic functionality is same

POSSIBLE QUESTIONS

PART A (20 x 1 = 20 marks)

(ONLINE EXAMINATION)

PART B (5 x 6 = 30 marks)

1. Describe the authentication dialogue used by Kerberos for obtaining required Services.
2. Explain in detail about IP security
3. Describe in detail about Electronic Mail Security.
4. Explain the services of PGP
5. Explain briefly about the PGP cryptographic function.
6. Explain in detail about the functions and requirements of S/MIME.

PART – C (1 x 10 = 10 marks)

1. Describe about X.509 services.
2. Explain the services of PGP
3. Explain in detail about the functions and requirements of S/MIME.
4. Explain in detail about IP security

KARPAGAM ACADEMY OF HIGHER EDUCATION						
DEPARTMENT OF COMPUTER SCIENCE						
I M.Sc CS						
CRYPTOGRAPHY AND NETWORK SECURITY						
	UNIT 5					
S.NO	Question	Option 1	Option 2	Option 3	Option 4	Answer
1	_____ is an authentication service designed for use in a distributed environment	kerberos	X.509	secure	none	kerberos
2	_____ defines the format for public key certificates	X.508	X.507	X.505	X.509	X.509
3	_____ is an authentication service developed as part of project athena at MIT	secure	X.509	kerberos	none	kerberos
4	A network eavesdropper should not be able to obtain the necessary information to impersonate a user is	reliable	secure	scalable	transparent	secure
5	_____,for all services that rely on kerberos for access control lack of availability of the service	reliable	secure	scalable	transparent	reliable
6	_____,the user should not be aware that authentication is taking place	reliable	transparent	secure	scalable	transparent
7	TGS stands for	ticket granting server	ticket gain server	ticket grant sell	track grant server	ticket granting server
8	_____ is a set of managed nodes that share the same kerberos database	secure	X.509	kerberos realm	none	kerberos realm
9	_____,which is a service or user that is known to the kerberos systems	kerberos principal	server	X.505	ticket granting server	kerberos principal
10	_____which provide unambiguous byte ordering	BER	AER	CER	DER	BER
11	BER stands for	basic enabling rules	basic emitting rules	basic editing rules	basic encoding rules	basic encoding rules
12	PCBC stands for	proper cipher block chaining	propagating cipher block chaining	propagating cipher base chaining	propagating chemical base chaining	propagating cipher block chaining
13	X.509 defines a frame work for the provision of authentication services by the directory to its	X.500	X.600	X.700	X.800	X.500
14	_____covers all of the other fields of the certificate	transparent	server	signature	secure	signature
15	_____ way authentication involves a single transfer of information from one user to another user	one	two	three	four	one
16	_____as the set of hardware,software,people policies and procedures	digital signature	private key infrastructure	public key infrastructure	none	public key infrastructure
17	_____used to denote any method for storing certificates and CRLs.	repository	private key infrastructure	public key infrastructure	none	repository
18	_____ is an open source freely available software package for e mail security	PGP	PGA	PGB	PGC	PGP
19	_____incorporates tools for developing a public key trust model and public key certificate management	PGA	PGB	PGP	PGC	PGP
20	_____ is an internet standard approach to e mail security that incorporates the same functionality as	S/MIME	PGP	digital signature	none	S/MIME
21	PGP stands for	prime good privacy	pretty good privacy	pretty good prime	pretty gain privacy	pretty good privacy
22	_____,which is provided by encrypting messages to be transmitted or to be stored locally as	PGP confidentiality	PGP compression	PGP segment	PGP email	PGP confidentiality
23	_____ the message after applying the signature but before encryption	PGP email	PGP segment	PGP compression	PGP confidentiality	PGP compression
24	_____ that indicates the extent to which PGP will trust that this is a valid public key for this user	signature trust key	owner trust key	key legitimacy field	none	key legitimacy field
25	_____ that indicates the degree to which this PGP users trust the signer to certify public keys	signature trust key	owner trust key	key legitimacy field	none	signature trust key
26	S/MIME stands for	secure/multipurpose internet mail	small/multipurpose internet mail extension	small/multiple internet mail expansion	secure/multipurpose internet message	secure/multipurpose internet mail extension
27	_____ defines a format for text message that are sent using electronic mail	RFC 822	RFC 833	RFC 922	RFC 722	RFC 822
28	_____ cannot transmit executable files or other binary objects	MMTP	MIME		SMME	SMTP
29	_____ cannot transmit text data that indicates national language characters	SMTP	MMTP	MIME	SMME	SMTP
30	_____ server may reject mail message over a certain size	MMTP	SMTP	MIME	SMME	SMTP
31	_____ used to identify MIME entities uniquely in multiple contexts	course id	server id	client id	content id	content id
32	_____ transfer encoding is useful when the data consists largely of octets that correspond to printable	server labels	security labels	quoted printable	client labels	quoted printable
33	_____ may be included in the authenticated attributes of a signed data object	security labels	quoted printable	server labels	client labels	security labels
34	_____ is a capability that can be added to either current version of the internet protocol	web security	IP security	internet security	network security	IP security
35	_____ defines a number of techniques for key management	IP security	internet security	network security	web security	IP security
36	_____ provides security services between TCP and application that use TCP	secure socket layer	handshake	transport layer service	cipher spec	secure socket layer
37	the internet standard version is called_____	handshake	transport layer service	secure socket layer	cipher spec	transport layer service
38	_____ is an open encryption and security specification designed to protect credit card	secure electronic transaction	handshake	cipher spec	none	secure electronic transaction
39	the most complex part of SSL is the _____ protocol	handshake	cipher spec	alter	none	handshake
40	_____ provides trust by the use of X.509v3 digital signature	handshake	cipher spec	secure electronic transaction	alter	secure electronic transaction
41	Which server acts as KDC in the Kerberos protocol	TGS	AS	Real server	None	AS
42	Which encryption algorithm is used in Kerberos 4 Protocol	AES	Block cipher	DES	Triple DES	DES
43	In which year was X.509 first issued?	1988	1978	1982	1994	1988
44	Who issues the PGP certificates?	ITU-T	IEEE	IETF	the user themselves	the user themselves
45	What are the two modes that IPSec protocol works on?	On and Off	Transport and tunnel	Forward and Backward	Linked and Unlinked	Transport and tunnel
46	Which of the following are IPSec protocols	PGP and S/MIME	Kerberos 4 , Kerberos 5	AH and ESP	SSL and SET	AH and ESP
47	IDEA stands for	International data esp algorithm	Internal data encryption algorithm	International data encryption analysis	International data encryption algorithm	International data encryption algorithm
48	DSS stands for	digital signature stream	data signature standard	digital signature standard	digital sign standard	digital signature standard

49	CA stands for	Certification Analysis	Certification Authorities	Certification AES	Certificate Authorities	Certification Authorities
50	Hash collision means	Two keys for one message	Two keys for one memory	Two keys for one member	Two keys for one merge	Two keys for one message
51	RSA stands for	Rivest,shamir,Addlem an	Rivest,sankar,Addlema n	Rivest,sankar,Addle	Rivest,san,Addleman	Rivest,shamir,Addlema n
52	KDC stands for	Key distribution center	Key display center	Key dynamic center	key dynamic cost	Key distribution center
53	SET provides an authentication with the help of _____.	digital certificate.	dual signature.	none	both	digital certificate.
54	The cryptography algorithms used in S/MIME are _____.	RSA,DES-3	RC4	RC5	RC6	RSA,DES-3
55	_____is the first step in DES.	Key transformation.	Expansion permutation.	both	none	Key transformation.
56	_____ refers more to asymmetric key cryptography.	Timing attack.	Meet in middle attack.	Virus attack.	Worms attack.	Timing attack.
57	The processed S/MIME along with security related data is called as _____.	public key cryptography	private key cryptography standard.	S/MIME	MIME	public key cryptography standard.
58	The physical form of money is converted into _____.	octal form.	hexadecimal.	decimal.	binary form.	binary form.
59	_____ cannot be used for encryption or key exchange	RSAA	AES	DES	RSA	RSAA
60	The art of breaking the code is_____	Cryptanalysis	AES	DES	RSA	Cryptanalysis