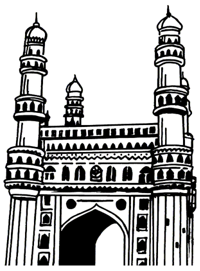
Rahul's ✔
Topper's Voice

NEW SYLLABUS



M.Sc.

(COMPUTER SCIENCE)

II Year III Sem

(Osmania University)

LATEST EDITION
2018 - 2019

NETWORK SECURITY

☞ **Study Manual**

☞ **Lab Programmes**

☞ **Solved Model Papers**

Price
~~`. 199-00~~
`. 149-00

- by -

WELL EXPERIENCED LECTURER

Rahul Publications™
Hyderabad. Ph : 66550071, 9391018098

M.Sc.

II Year III Sem

NETWORK SECURITY

☞ **Study Manual**

☞ **Lab Programmes**

☞ **Solved Model Papers**

Price
`. ~~199-00~~
`. 149-00

Sole Distributors : ☎ : 66550071, Cell : 9391018098

VASU BOOK CENTRE

Shop No. 3, Beside Gokul Chat, Koti, Hyderabad.

Maternity Hospital Opp. Lane, Narayan Naik Complex, Koti, Hyderabad.
Near Andhra Bank, Subway, Sultan Bazar, Koti, Hyderabad -195.

CONTENTS

NETWORK SECURITY

SYLLABUS

UNIT – I

Overview of Network Security: Computer Security Concepts,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. Block Ciphers and the Data Encryption Standard:Traditional Block Cipher Structure, the Data Encryption Standard (DES), A DES Example, Strength of DES.Block Cipher Operation: Double DES, Triple DES, Electronic Code Book, Cipher Block Chaining Mode, Cipher Feedback Mode, Output Feedback Mode, Counter Mode.

UNIT – II

Advanced Encryption Standard (AES): The Origins AES, AES Structure, AES Round Functions, AES Key Expansion, an AES Example AES Implementation.Pseudorandom Number Generation and Stream Ciphers: Principles of Pseudorandom Number Generation, Pseudorandom Number Generators,Pseudorandom Number Generation using BlockCipher, StreamCiphers-RC4. Public-Key Cryptography and RSA: Principles of Public-Key Cryptosystems, the RSA Algorithm.Key Management and Distribution: Symmetric Key Distribution Using Symmetric Encryption and Asymmetric Encryption, Distribution of Public Keys, X.509 Certificates, Diffie-Hellman Key Exchange.

UNIT – III

Cryptographic Hash Functions: Applications of Cryptographic Hash Functions, Two Simple Hash Functions, Secure Hash Algorithm (SHA) & MD5 Algorithm.Message Authentication Codes: Message Authentication Requirements, Message Authentication Functions, Requirements for Message Authentication Codes,Security of MACs,MACs Based on Hash Functions:HMAC,MACs Based on Block Ciphers:DAA and CMAC.Digital Signatures: Digital Signatures, NIST Digital Signatures Algorithm.

UNIT – IV

Transport-Level Security : Web Security Considerations, Secure Sockets Layer (SSL), Transport Layer Security (TLS), HTTPS,Secure Shell (SSH), E-Mail Security: Pretty Good Privacy, S/MIME. IP Security: IP Security Overview, IP Security Architecture, Encapsulating Security Payload, Combining Security Associations,Internet Key Exchange.Intruders, Virus and Firewalls: Intruders, Intrusion Detection, Password Management, Virus and Related Threats, Countermeasures, Firewall Design Principles, Types of Firewalls.

Contents

Topic No. **Page No.**

UNIT I

Overview of Network Security: Computer Security Concepts,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. Block Ciphers and the Data Encryption Standard:Traditional Block Cipher Structure, the Data Encryption Standard (DES), A DES Example, Strength of DES.Block Cipher Operation: Double DES, Triple DES, Electronic Code Book, Cipher Block Chaining Mode, Cipher Feedback Mode, Output Feedback Mode, Counter Mode.

1.1 Overview of Network Security

1.1.1 Computer Security Concepts

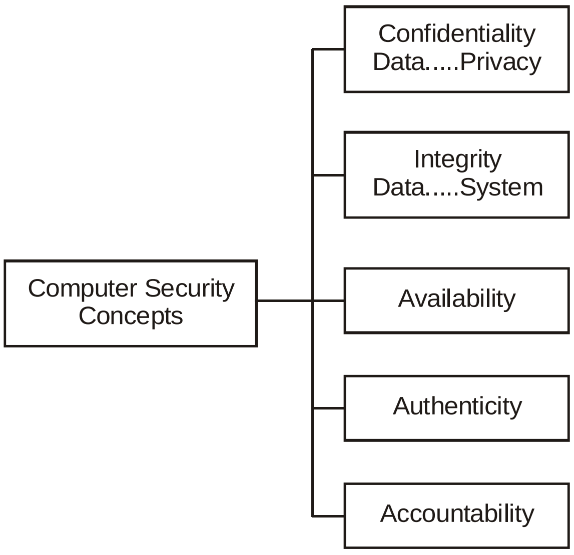
Q1. What is Computer Security ?

Ans :

Computer security is refers to techniques for ensuring that data stored in a computer cannot be read or compromised by any individuals without authorization.

- Most computer security measures involve data encryption and passwords.
- The purpose of computer security is to device ways to prevent the weaknesses from being,

There are five important aspects of any computer-related system such as confidentiality, integrity, and availability, authenticity and accountability.


Computer Security Concepts
Confidentiality Data.....Privacy
Integrity Data.....System
Availability
Authenticity
Accountability

Confidentiality

- **Data confidentiality**: Assures that confidential information is not disclosed to unauthorized individuals
- **Privacy**: Assures that individual control or influence what information may be collected and stored

Integrity

- **Data integrity**: assures that information and programs are changed only in a specified and authorized manner
- **System integrity**: Assures that a system performs its operations in unimpaired manner

Availability

Assure that systems works promptly and service is not denied to authorized users

- **Authenticity**: the property of being genuine and being able to be verified and trusted; confident in the validity of a transmission, or a message, or its originator

Accountability

Generates the requirement for actions of an entity to be traced uniquely to that individual to support nonrepudiation, deference, fault isolation, etc.

1.1.2 The OSI Security Architecture

Q2. What is the OSI security architecture ?

Ans :

The OSI Security Architecture is a framework that provides a systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. The document defines security attacks, mechanisms, and services, and the relationships among these categories.

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

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

  There are four general categories of attack which are listed below.

- **Interruption**
  - Interception
  - Modification
  - Fabrication

  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.

Active attacks

These attacks involve some modification of the data stream or the creation of a false stream.

- **Security mechanism**: A process (or a device incorporating such a process) thatis designed to detect, prevent, or recover from a security attack.

  Some of the mechanisms are:

  - Encipherment
  - Digital Signature
  - Access Control

- Security service: A processing or communication service that enhances thesecurity of the data processing systems and the information transfers of anorganization. The services are intended to counter security attacks, and theymake use of one or more security mechanisms to provide the service.

The classification of security services are as follows:

- Confidentiality
- Authentication
- Integrity
- Non repudiation
- Access control
- Availability

1.1.3 Security Attack

Q3. Explain about various security attacks in network

Ans :

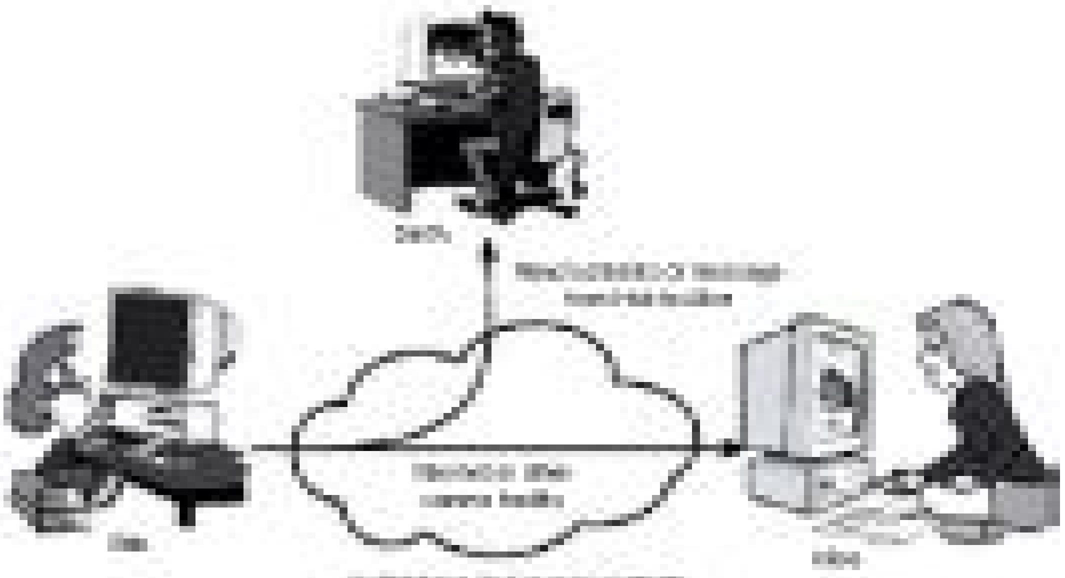
A useful means of classifying security attacks,is in terms of *passive attacks* and *active attacks.* A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

Passive Attacks

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

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

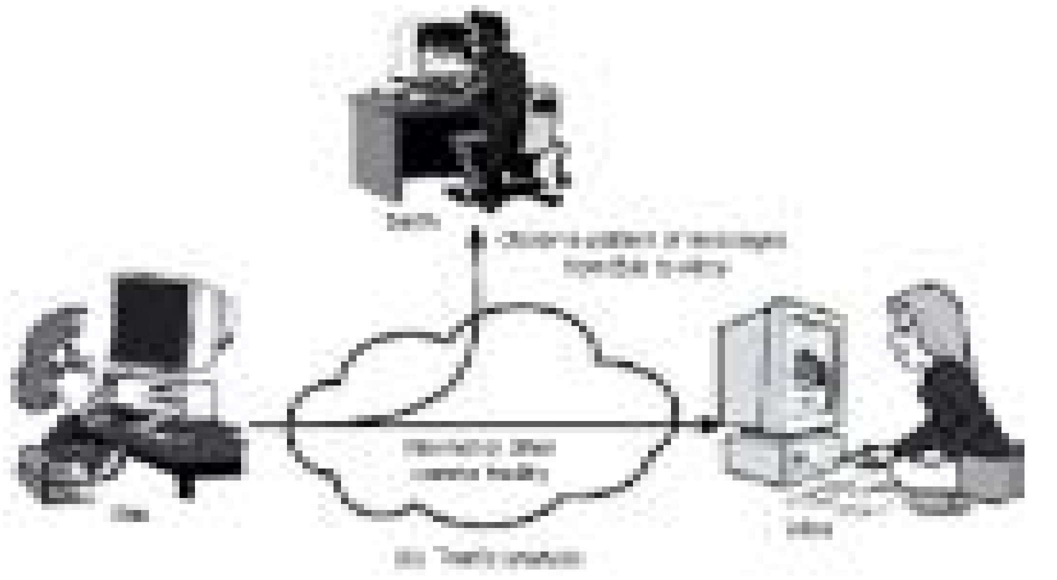
The release of message contents is easily understood (a). A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.


Darth
Read contents of message from Bob to Alice
Internet or other communications facility
Bob
Alice

(a) Release of message contents

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

Passive attacks are very difficult to detect, because they do not involve any alteration of the data. Typically, the message traffic is sent and received in an apparently normal fashion, and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern. However, it is feasible to pre-vent the success of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

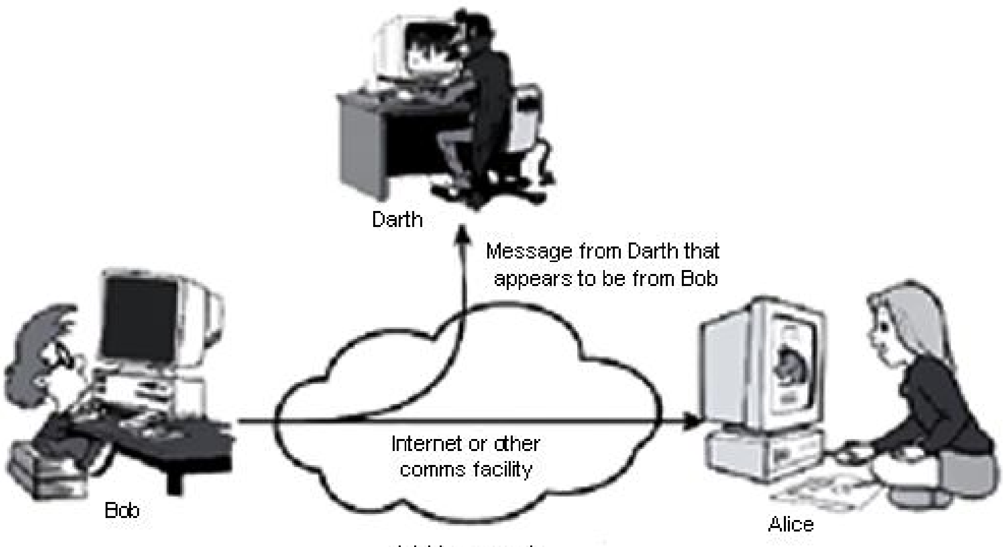

Darth
Observe pattern of messages from Bob to Alice
Internet or other communications facility
Bob
Alice

(b) Traffic analysis

Active Attacks

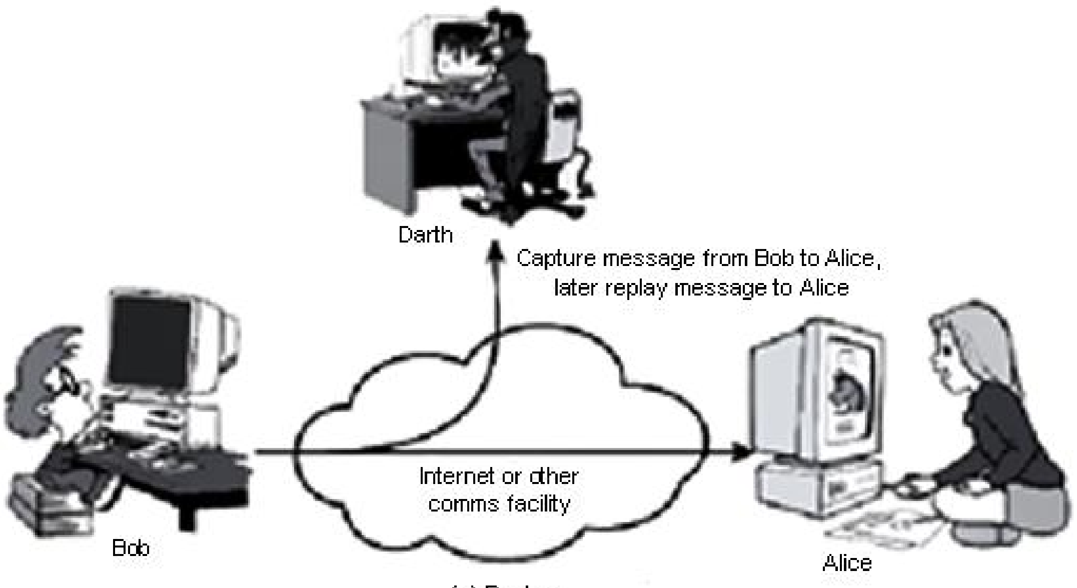
Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories:

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


Darth
Message from Darth that appears to be from Bob
Internet or other comms facility
Bob
Alice

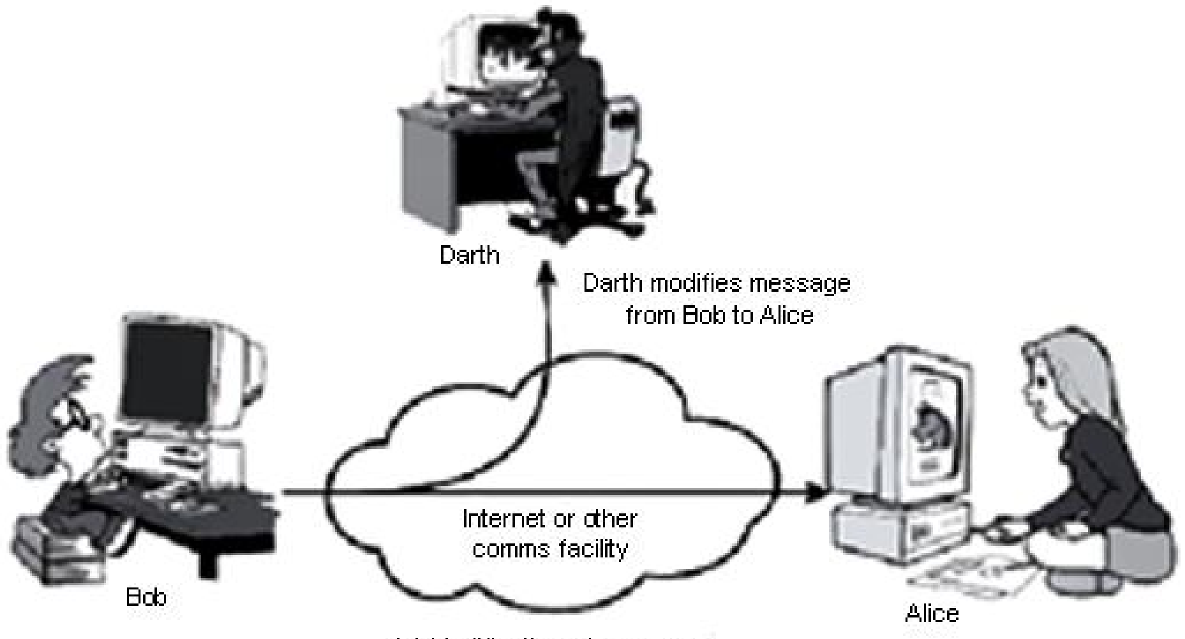
(a) Masquerade

Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect (Figure 1.3b).


Darth
Capture message from Bob to Alice, later replay message to Alice
Internet or other comms facility
Bob
Alice

(a) Replay

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


Darth
Darth modifies message from Bob to Alice
Internet or other comms facility
Bob
Alice
(a) Modification of messages

The **denial of service** prevents or inhibits the normal use or management of communications facilities (Figure 1.3d). This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.

Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success.

On the other hand, it is quite difficult to prevent active attacks absolutely because of the wide variety of potential physical, software, and network vulnerabilities. Instead, the goal is to detect active attacks and to recover from any disruption or delays caused by them. If the detection has a deterrent effect, it may also contribute to prevention.

1.1.4 Security Services

Q4. Explain about various Security Services in Network.

Ans :

It is a processing or communication service that is provided by a system to give a specific kind of production to system resources. Security services implement security policies and are implemented by security mechanisms.

- **Confidentiality**

  Confidentiality is the protection of transmitted data from passive attacks. It is used to prevent the disclosure of information to unauthorized individuals or systems. It has been defined as "ensuring that information is accessible only to those authorized to have access" .The other aspect of confidentiality is the protection of traffic flow from analysis. **Ex:** A credit card number has to be secured during online transaction.

- **Authentication**

  This service assures that a communication is authentic. For a single message transmission, its function is to assure the recipient that the message is from intended source. For an ongoing interaction two aspects are involved. First, during connection initiation the service assures the authenticity of both parties. Second, the connection between the two hosts is not interfered allowing a third party to masquerade as one of the two parties. Two specific authentication services defines in X.800 are

  - **Peer entity authentication:** Verifies the identities of the peer entities involved in communication. Provides use at time of connection establishment and during data transmission. Provides confidence against a masquerade or a replay attack

  - **Data origin authentication:** Assumes the authenticity of source of data unit, but does not provide protection against duplication or modification of data units. Supports applications like electronic mail, where no prior interactions take place between communicating entities.

- **Integrity**

  Integrity means that data cannot be modified without authorization. Like confidentiality, it can be applied to a stream of messages, a single message or selected fields within a message. Two types of integrity services are available. They are

- **Connection-Oriented Integrity Service** This service deals with a stream of messages, assures that messages are received as sent, with no duplication, insertion, modification, reordering or replays. Destruction of data is also covered here. Hence, it attends to both message stream modification and denial of service.

- **Connectionless-Oriented Integrity Service:** It deals with individual messages regardless of larger context, providing protection against message modification only.

An integrity service can be applied with or without recovery. Because it is related to active attacks, major concern will be detection rather than prevention. If a violation is detected and the service reports it, either human intervention or automated recovery machines are required to recover.

- **Non-repudiation**

  Non-repudiation prevents either sender or receiver from denying a transmitted message. This capability is crucial to e-commerce. Without it an individual or entity can deny that he, she or it is responsible for a transaction, therefore not financially liable.

- **Access Control**

  This refers to the ability to control the level of access that individuals or entities have to a network or system and how much information they can receive. It is the ability to limit and control the access to host systems and applications via communication links. For this, each entity trying to gain access must first be identified or authenticated, so that access rights can be tailored to the individuals.

- **Availability**

  It is defined to be the property of a system or a system resource being accessible and usable upon demand by an authorized system entity. The availability can significantly be affected by a variety of attacks, some amenable to automated counter measures i.e authentication and encryption and others need some sort of physical action to prevent or recover from loss of availability of elements of a distributed system.

1.1.5 Security Mechanisms

Q5. Explain about various Security Mechanisms in Network.

Ans :

The security mechanisms are divided into those implemented in a specific protocol layer and those that are not specific to any particular protocol layer or security service. Also differentiates reversible & irreversible encipherment mechanisms. A reversible encipherment mechanism is simply an encryption algorithm that allows data to be encrypted and subsequently decrypted, whereas irreversible encipherment include hash algorithms and message authentication codes used in digital signature and message authentication applications.

- **Specific Security Mechanisms**

  Incorporated into the appropriate protocol layer in order to provide some of the OSI security services,

- **Encipherment:** It refers to the process of applying mathematical algorithms for converting data into a form that is not intelligible. This depends on algorithm used and encryption keys.
- **Digital Signature:** The appended data or a cryptographic transformation applied to any data unit allowing to prove the source and integrity of the data unit and protect against forgery.
- **Access Control:** A variety of techniques used for enforcing access permissions to the system resources.
- **Data Integrity:** A variety of mechanisms used to assure the integrity of a data unit or stream of data units.
- **Authentication Exchange:** A mechan ism 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 once a breach of security is suspected.
- **Notarization:** The use of a trusted third party to assure certain properties of a data exchange
- **Pervasive Security Mechanisms**

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

- **Trusted Functionality:** That which is perceived to b correct with respect to some criteria
- **Security Level:** The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
- **Event Detection:** It is the process of detecting all the events related to network security.
- **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:** It deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

1.1.6 A Model For Network Security

Q6. Explain about a Model for Network Security.

Ans :

A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender.

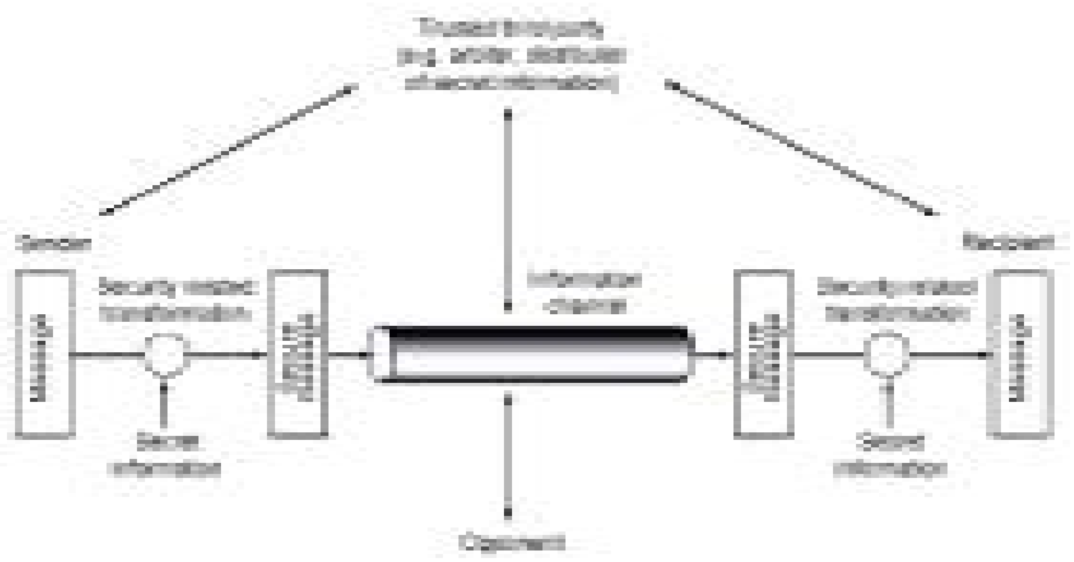


Fig. : Model for Network Security

Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.

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

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

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

   (e.g., obtaining credit card numbers or performing illegal money transfers).

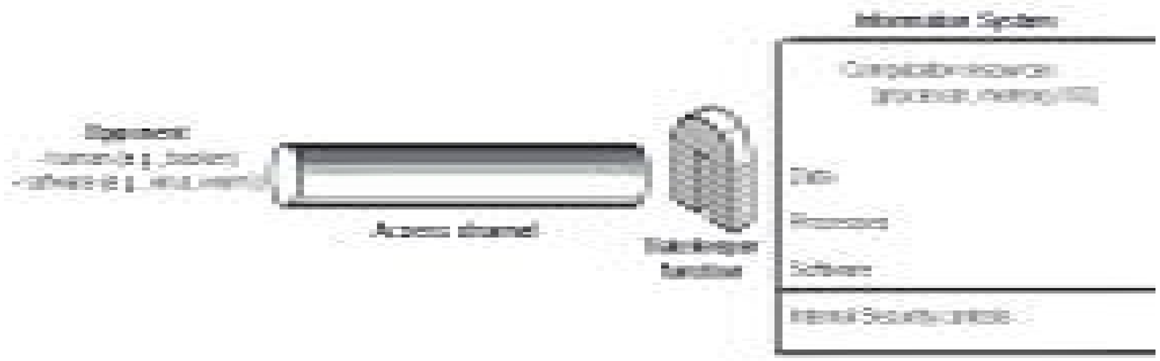


Fig. : Network Access Security Model

Another type of unwanted access is the placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application pro-grams as well as utility programs, such as editors and compilers. Programs can present two kinds of threats:

Information access threats

Intercept or modify data on behalf of users who should not have access to that data.

Service threats

Exploit service flaws in computers to inhibit use by legitimate users.

Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a disk that contains the unwanted logic concealed in otherwise useful software. They can also be inserted into a system across a network; this latter mechanism is of more concern in network security.

1.2 Classical Encryption Techniques

Q7. Write about the various terminologies used in classical Encryption Techniques.

Ans :

- **Plaintext**: original message
- **Ciphertext**: coded message
- **Enciphering** or **encryption**: the process of converting from plaintext to ciphertext
- **Deciphering** or **decryption:** the process of restoring the plaintext from the ciphertext

The many schemes used for encryption constitute the area of study known as cryptography. Such a scheme is known as a cryptographic system (cryptosystem) or a cipher. Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of cryptanalysis. Cryptanalysis is what the layperson calls "breaking the code". The areas of cryptography and cryptanalysis together are called cryptology.

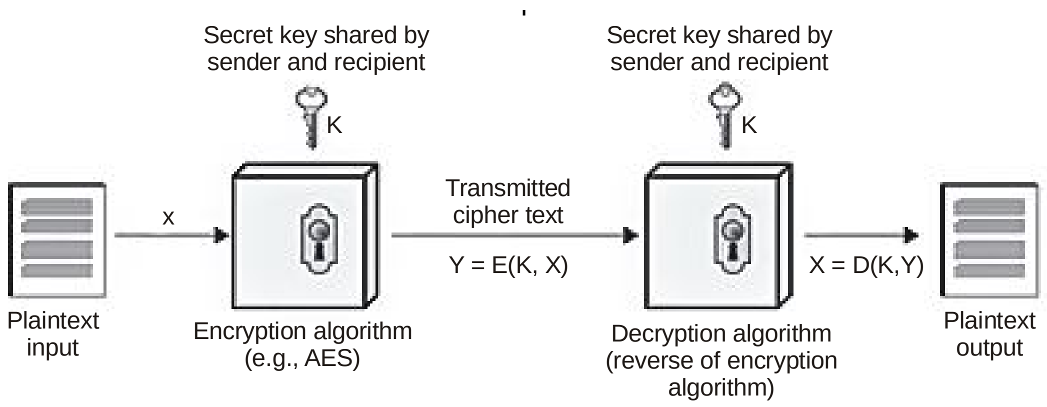
1.2.1 Symmetric Cipher Model

Q8. Write about the Symmetric Cipher Model.

Ans :

A symmetric encryption scheme has five ingredients (as shown in the following figure):

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


Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
x
Transmitted cipher text
Y = E(K, X)
X = D(K,Y)
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Encryption Requirements

There are two requirements for secure use of conventional encryption:

1. The encryption algorithm must be strong.

   At a minimum, an opponent who knows the algorithm and has access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key.

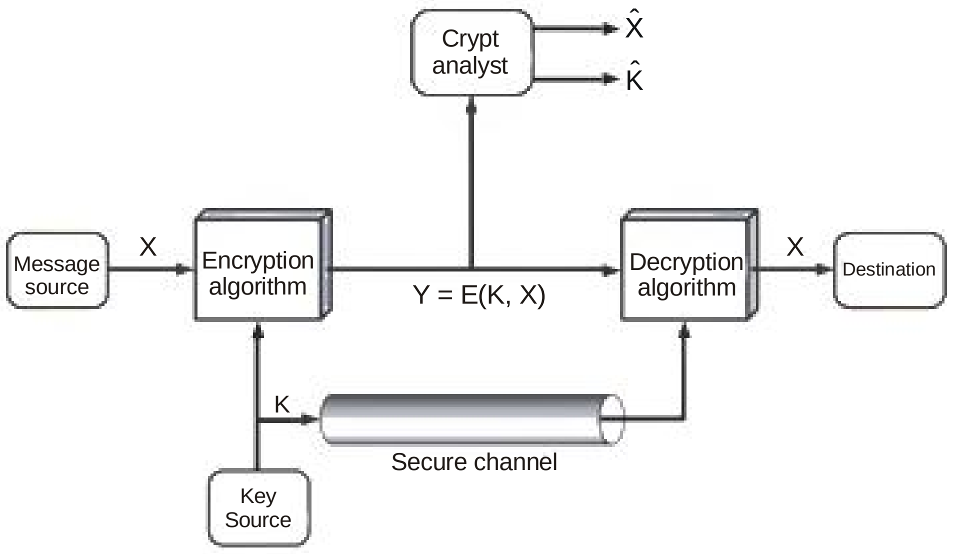
   In a stronger form, the opponent should be unable to decrypt ciphertexts or discover the key even if he or she has a number of ciphertexts together with the plaintext for each ciphertext.

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

We assume that it is impractical to decrypt a message on the basis of the ciphertext plus knowledge of the encryption/decryption algorithm. This means we do not need to keep the algorithm secret; we need to keep only the key secret. This feature of symmetric encryption makes low-cost chip implementations of data encryption algorithms widely available and incorporated into a number of products. With the use of symmetric encryption, the principal security problem is maintaining the secrecy of the key.

Model of Symmetric Cryptosystem

The essential elements of a symmetric encryption scheme is described in the following figure:



- A source produces a message in plaintext, 'X=[X1,X2,...,XM]
- A key of the form K=[K1,K2,...,KJ] is generated.
  - If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel.
  - Alternatively, a third party could generate the key and securely deliver it to both source and destination.
- The ciphertext Y=[Y1,Y2,...,YN] is produced by the encryption algorithm with the message *X* and the encryption key *K*as input.

The encryption process is

Y=E(K,X)

This notation indicates that *Y* is produced by using encryption algorithm E as a function of the plaintext *X*, with the specific function determined by the value of the key *K*.

The intended receiver with the key is able to invert the transformation:

X=D(K,Y)

An opponent, observing *Y* but not having access to *K* or *X*, may attempt to recover *X* or *K* or both. It is assumed that the opponent knows the encryption (E) and decryption (D) algorithms. The opponent may do one of the following:

- Recover *X* by generating a plaintext estimate X^, if the opponent is interested in only this particular message.
- Recover *K* by generating an estimate K^, if the opponent is interested in being able to read future messages.

1.2.2 Substitution Techniques

Q9. Explain about various substitution techniques of encryption.

Ans :

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

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.

- **CAESAR CIPHER**

The earliest known use of a substitution cipher and the simplest was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing 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) mod 26

A shift may be any amount, so that general Caesar algorithm is C = E (p) = (p+k) mod 26

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

The decryption algorithm is simply P = D(C) = (C-k) mod 26

➢ MONOALPHABETIC CIPHERS

Here, Plaintext characters are substituted by a different alphabet stream of characters shifted to the right or left by n positions. When compared to the Caesar ciphers, these monoalphabetic ciphers are more secure as each letter of the ciphertext can be any permutation of the 26 alphabetic characters leading to 26! or greater than 4 x 1026 possible keys. But it is still vulnerable to cryptanalysis, when a cryptanalyst is aware of the nature of the plaintext, he can find the regularities of the language. To overcome these attacks, multiple substitutions for a single letter are used. For example, a letter can be substituted by different numerical cipher symbols such as 17, 54, 69….. etc. Even this method is not completely secure as each letter in the plain text affects on letter in the ciphertext.

Or, using a common key which substitutes every letter of the plain text.

The key ABCDEFGHIIJ KLMNOPQRSTUVWXYZ

QWERTYUIIOPAS DFGHJ KLZXCV BNM

Would encrypt the message

II think therefore II am

into

OZIIOFAZIITKTYGKTOQD

But any attacker would simply break the cipher by using frequency analysis by observing the number of times each letter occurs in the cipher text and then looking upon the English letter frequency table. So, substitution cipher is completely ruined by these attacks.

➢ PLAYFAIR CIPHERS

It is the best known multiple –letter encryption cipher which treats digrams in the plaintext as single units and translates these units into ciphertext digrams. The Playfair Cipher is a digram substitution cipher offering a relatively weak method of encryption. It is based around a 5x5 matrix, a copy of which is held by both communicating parties, into which 25 of the 26 letters of the alphabet (normally either j and i are represented by the same letter or x is ignored) are placed in a random fashion. For example, the plain text is *Shi Sherry loves Heath Ledger* and the agreed key is *sherry*. The matrix will be built according to the following rules.

- ➢ in pairs,
- ➢ without punctuation,
- ➢ All Js are replaced with Is.

SH IS HE RR YL OV ES HE AT HL ED GE R

- ➢ Double letters which occur in a pair must be divided by an X or a Z.

E.g. ***LI TE RA LL YLI TE RA LX LY***

- ➢ *SH IS HE RX RY LO VE SH EA TH LE DG ER* The alphabet square is prepared using, a 5*5 matrix, no repetition letters, no Js and key is written first followed by the remaining alphabets with no i and j.

S H E R Y

A B C D F

G I K L M

N O P Q T

U V W X Z

- ➢ For the generation of cipher text, there are three rules to be followed by each pair of letters.
- ➢ letters appear on the same rowÿreplace them with the letters to their immediate right respectively
- ➢ letters appear on the same columnÿreplace them with the letters immediately below respectively
- ➢ not on the same row or columnÿreplace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair.

- Based on the above three rules, the cipher text obtained for the given plain text is

HE GH ER DR YS IQ WH HE SC OY KR AL RY

Another example which is simpler than the above one can be given as:

Here, key word is *playfair*. Plaintext is *Hellothere hellothere* becomes——*-he lx lo th er ex* . Applying the rules again, for each pair, If they are in the same row, replace each with the letter to its right (mod 5)

He = = KG

If they are in the same column, replace each with the letter below it (mod 5)

Lo = = V

Otherwise, replace each with letter we'd get if we swapped their column indices

lx = YV

So the cipher text for the given plain text is **KG YV RV QM GI KU**

$$\begin{bmatrix} p & l & a & y & f \\ i & r & b & c & d \\ e & g & h & k & m \\ n & o & q & s & t \\ u & v & w & x & z \end{bmatrix}$$

To decrypt the message, just reverse the process. Shift up and left instead of down and right. Drop extra x's and locate any missing I's that should be j's. The message will be back into the original readable form.

- **HILL CIPHER**

It is also a multi letter encryption cipher. It involves substitution of '*m*' ciphertext letters for '*m*' successive plaintext letters. For substitution purposes using '*m*' linear equations, each of the characters are assigned a numerical values i.e. a=0, b=1, c=2, d=3,…….z=25. For example if m=3, the system can be defined as: **c1 = (k11p1 + k12p2 + k13p3) mod 26 c2 = (k21p1 + k22p2 + k23p3) mod 26 c3 = (k31p1 + k32p2 + k33p3) mod 26** If we represent in matrix form, the above statements as matrices and column vectors:

$$\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \text{mod } 2/S$$

Thus, C = KP *mod26,* where C= Column vectors of length 3 P = Column vectors of length 3 K = 3x3 encryption key matrix. For decryption process, inverse of matrix **K** i.e. **K-1** is required which is defined by the equation KK-1 = K-1K = I, where I is the identity matrix that contains only 0's and 1's as its elements. Plaintext is recovered by applying K-1 to the cipher text. It is expressed as C = EK(P) = KP *mod26* P = DK(C) = K-1C *mod26.* = K-1KP = IP = P

Example: The plain text is I can't do it and the size of m is 3 and key K is chosen as following

I Can't do it

8 2 0 13 19 3 14 8 19

$$\begin{pmatrix} 9 & 18 & 10 \\ 16 & 21 & 1 \\ 5 & 12 & 23 \end{pmatrix}$$

The encryption process is carried aout as follows

$$\begin{pmatrix} 4 \\ 14 \\ 12 \end{pmatrix} = \begin{pmatrix} 9 & 18 & 10 \\ 16 & 21 & 1 \\ 5 & 12 & 23 \end{pmatrix} \begin{pmatrix} 8 \\ 2 \\ 0 \end{pmatrix} (\text{mod } 26)$$

$$\begin{pmatrix} 19 \\ 12 \\ 14 \end{pmatrix} = \begin{pmatrix} 9 & 18 & 10 \\ 16 & 21 & 1 \\ 5 & 12 & 23 \end{pmatrix} \begin{pmatrix} 13 \\ 19 \\ 3 \end{pmatrix} (\text{mod } 26)$$

$$\begin{pmatrix} 8 \\ 21 \\ 9 \end{pmatrix} = \begin{pmatrix} 9 & 18 & 10 \\ 16 & 21 & 1 \\ 5 & 12 & 23 \end{pmatrix} \begin{pmatrix} 14 \\ 8 \\ 19 \end{pmatrix} (\text{mod } 26)$$

So, the encrypted text will be given as $\rightarrow$ EOM TMY SV1

The main advantages of hill cipher are given below:

- It perfectly hides single-letter frequencies.
- Use of **3x3** Hill ciphers can perfectly hide both the single letter and two-letter frequency information.
- Strong enough against the attacks made only on the cipher text.

But, it still can be easily broken if the attack is through a known plaintext.

1.2.3 Transposition Techniques

Q10. Write about various transposition techniques.

Ans :

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.

Rail fence 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 at the school house

To encipher this message with a rail fence of depth 2,

We write the message as follows :

meatecolosetthshhue.

The encrypted message is M E A T E C O L O S E T T H S H O H U E

Row Transposition Ciphers - A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of columns then becomes the key of the algorithm.

e.g., plaintext = meet at the school house

Key = 4 3 1 2 5 6 7

PT = m e e t a t t h e s c h o o l h o u s e

CT = ESOTCUEEHMHLAHSTOETO

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.

1.2.4 Rotor MAchines

Q11. What are rotor machines? Explain.

Ans :

The basic principle of the rotor machine is illustrated in Figure 2.8. The machine consists of a set of independently rotating cylinders through which electri-cal 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 2.8, 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 poly-alphabetic substitution algorithm with a period of 26.

A single-cylinder system is trivial and does not present a formidable cryptana-lytic 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 2.8 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 2.8 shows the system's config-uration 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 * 26 * 26 = 17,576 different substitution alphabets used before the system repeats. The addition of fourth and fifth rotors results in periods of 456,976 and 11,881,376 letters, respectively. As David Kahn eloquently put it, referring to a five-rotor machine [KAHN96,]:

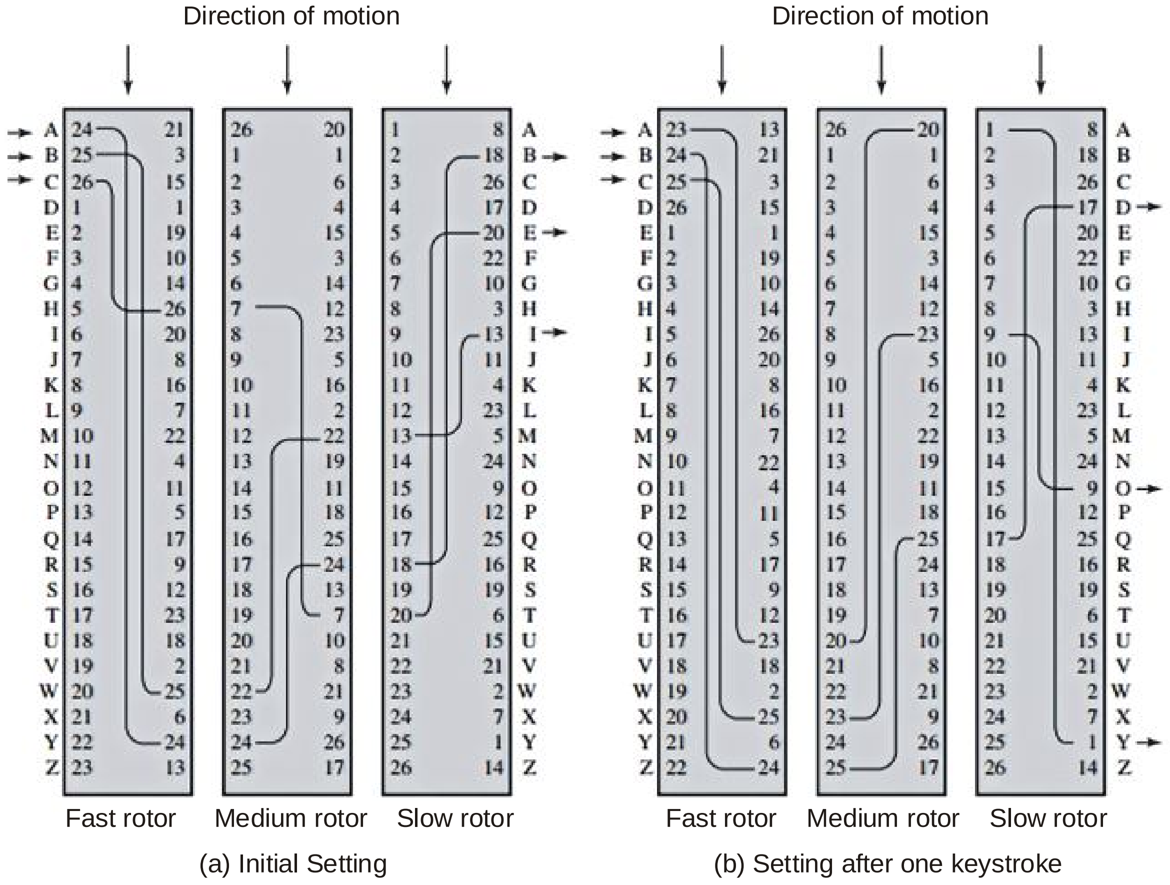


Fig. : Three-Rotor Machine with Wiring Represented by Numbered Contacts

A period of that length thwarts any practical possibility of a straightforward solution on the basis of letter frequency. This general solution would need about 50 letters per cipher alphabet, meaning that all five rotors would have to go through their com-bined cycle 50 times. The ciphertext would have to be as long as all the speeches made on the floor of the Senate and the House of Representatives in three successive sessions of Congress. No cryptanalyst is likely to bag that kind of trophy in his lifetime; even diplomats, who can be as verbose as politicians, rarely scale those heights of loquacity.

1.2.5 Steganography

Q12. What is stenography? Write about it.

Ans :

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, but one that is time consuming to construct is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message. 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.

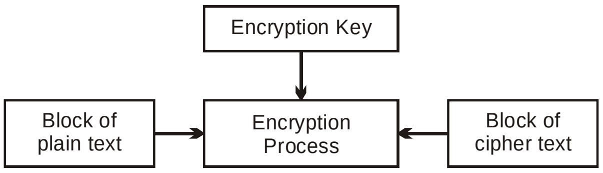
1.3 Block Ciphers and the Data Encryption Standard

Q13. What is block cipher ?

Ans :

A block cipher is a method of encrypting text (to produce ciphertext) in which acrypto graphic key and algorithm are applied to a block of data (for example, 64 contiguous bits) at once as a group rather than to one bit at a time. The main alternative method, used much less frequently, is called the stream cipher.

The basic scheme of a block cipher is depicted as follows :



A block cipher takes a block of plaintext bits and generates a block of ciphertext bits, generally of same size. The size of block is fixed in the given scheme. The choice of block size does not directly affect to the strength of encryption scheme. The strength of cipher depends up on the key length.

1.3.1 Traditional Block Cipher Structure

Q14. Explain about Traditional Block Cipher Structure.

Ans :

Though any size of block is acceptable, following aspects are borne in mind while selecting a size of a block.

- **Avoid very small block size** – Say a block size is m bits. Then the possible plaintext bits combinations are then 2^m. If the attacker discovers the plain text blocks corresponding to some previously sent ciphertext blocks, then the attacker can launch a type of 'dictionary attack' by building up a dictionary of plaintext/ ciphertext pairs sent using that encryption key. A larger block size makes attack harder as the dictionary needs to be larger.
- **Do not have very large block size** – With very large block size, the cipher becomes inefficient to operate. Such plaintexts will need to be padded before being encrypted.
- **Multiples of 8 bit** – A preferred block size is a multiple of 8 as it is easy for implementation as most computer processor handle data in multiple of 8 bits.

Padding in Block Cipher

Block ciphers process blocks of fixed sizes (say 64 bits). The length of plaintexts is mostly not a multiple of the block size. For example, a 150-bit plaintext provides two blocks of 64 bits each with third block of balance 22 bits. The last block of bits needs to be padded up with redundant information so that the length of the final block equal to block size of the scheme. In our example, the remaining 22 bits need to have additional 42 redundant bits added to provide a complete block. The process of adding bits to the last block is referred to as padding.

Too much padding makes the system inefficient. Also, padding may render the system insecure at times, if the padding is done with same bits always.

Block Cipher Schemes

There is a vast number of block ciphers schemes that are in use. Many of them are publically known. Most popular and prominent block ciphers are listed below.

- **Digital Encryption Standard (DES)** " The popular block cipher of the 1990s. It is now considered as a 'broken' block cipher, due primarily to its small key size.
- **Triple DES** – It is a variant scheme based on repeated DES applications. It is still a respected block ciphers but inefficient compared to the new faster block ciphers available.
- **Advanced Encryption Standard (AES)** – It is a relatively new block cipher based on the encryption algorithm Rijndael that won the AES design competition.
- **IDEA** – It is a sufficiently strong block cipher with a block size of 64 and a key size of 128 bits. A number of applications use IDEA encryption, including early versions of Pretty Good Privacy (PGP) protocol. The use of IDEA scheme has a restricted adoption due to patent issues.
- **Twofish** – This scheme of block cipher uses block size of 128 bits and a key of variable length. It was one of the AES finalists. It is based on the earlier block cipher Blowfish with a block size of 64 bits.
- **Serpent** – A block cipher with a block size of 128 bits and key lengths of 128, 192, or 256 bits, which was also an AES competition finalist. It is a slower but has more secure design than other block cipher.

1.3.2 The Data Encryption Standard

Q15. Explain about DES algorithm.

Ans :

The most widely used encryption scheme is based on the Data Encryption Standard (DES). For DES, data are encrypted in 64-bit blocks using a 56- bit key. The algorithmtransforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used toreverse the encryption.

DES Encryption

The overall scheme for DES encryption is illustrated in figure 3.5. As with any encryption scheme, thereare two inputs to the encryption function: the plaintext to be encrypted and the key. In this case, the plaintextmust be 64 bits in length and the key is 56 bits in length.

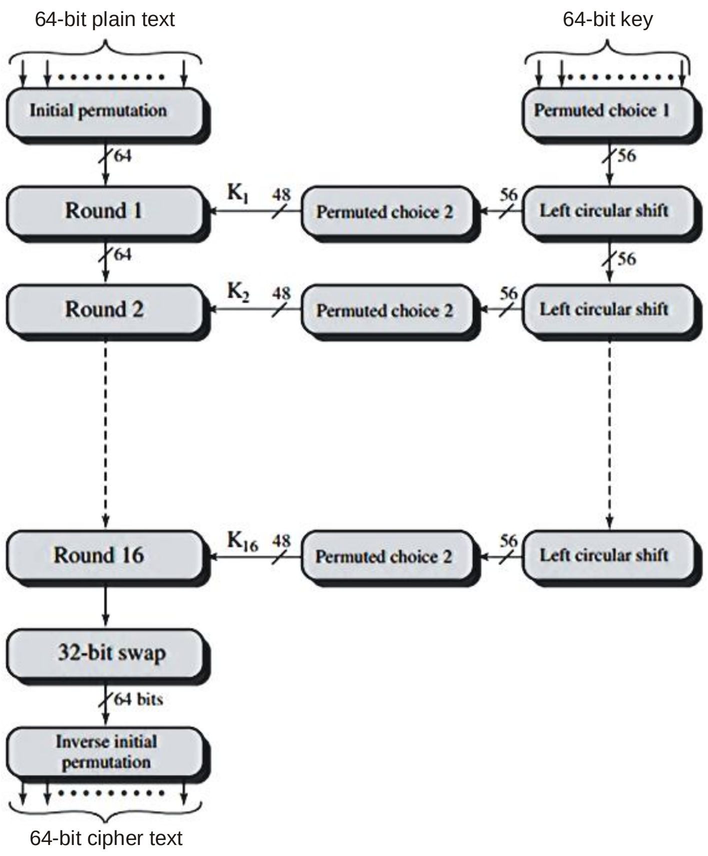


Fig. : General Depiction of DES Encryption Algorithm

Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases. First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input. This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left andright halves of the output areswapped to produce the preoutput. DES has the exact structure of a Feistel cipher, as shown in Figure 3.3.

The right-hand portion of Figure 3.5 shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the sixteen rounds, a subkey (Ki) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

Initial Permutation

The initial permutation and its inverse are defined by tables, as shown in Tables 3.2a and 3.2b, respectively. The tables are to be interpreted as follows. The input to a table consists of 64 bits numbered from 1 to 64. The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64. Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits.

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

(a) Initial Permutation (IP)

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

(b) Inverse Initial Permutation (IP)

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

(c) Expansion Permutation (E)

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

(d) Permutation Function (P)

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

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

where M_i is a binary digit. Then the permutation $X = (\mathrm{IP}(M)$ is as follows:

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

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

DETAILS OF SINGLE ROUND Figure 3.6 shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas:

$$Li = Ri - 1$$

$$Li = Ri - 1$$

$$Ri = L_{i-1} + F(R_{i-1}, K_i)$$

The round key Ki is 48 bits. The R input is 32 bits. This R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the R bits (Table 3.2c). The resulting 48 bits are XORed with Ki . This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as defined by Table 3.2d.

The role of the S-boxes in the function F is illustrated in Figure 3.7. The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. These transformations aredefined in Table 3.3, which isinterpreted as follows: The first and last bits of the input to box Si form a 2-bit binary number to select one offour substitutions defined by the four rows in the table for Si. The middle four bits select one of the sixteencolumns. The decimal value in the cell selected by the row and column is then converted to its 4-bitrepresentation to pro- duce the output. For example, in S1, for input 011001, the row is 01 (row 1) and thecolumn is 1100 (column 12). The value in row 1, column 12 is 9, so the output is 1001.

Each row of an S-box defines a general reversible substitution. Figure 3.2 may be useful in understanding themapping. The figure shows the substitution for row 0 of box S1.

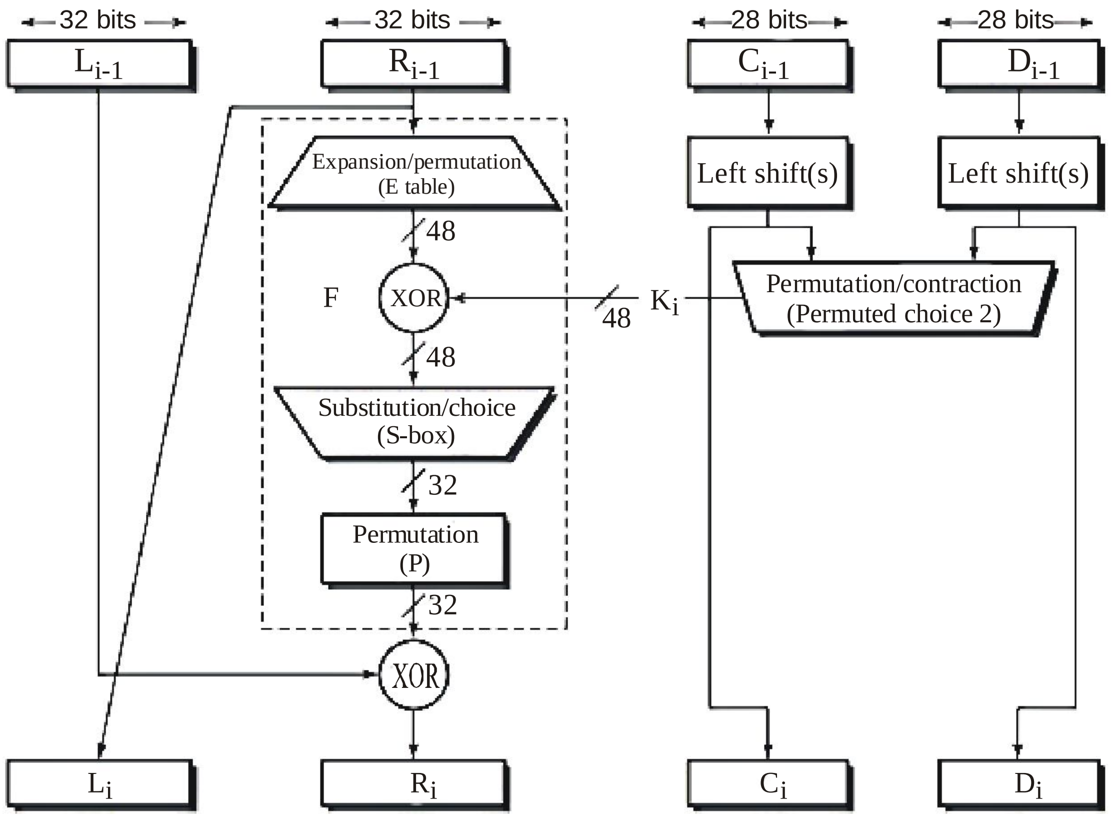


Fig. : Single Round of DES Algorithm

The operation of the S-boxes is worth further comment. Ignore for the momentthe contribution of the key (Ki). If you examine the expansion table, you see that the 32 bits of input are splitinto groups of 4 bits and then become groups of 6 bits by taking the outer bits from the two adjacent groups. Forexample, if part of the input word is

... efgh ijkl mnop ...

this becomes ... defghi hijklm lmnopq ...

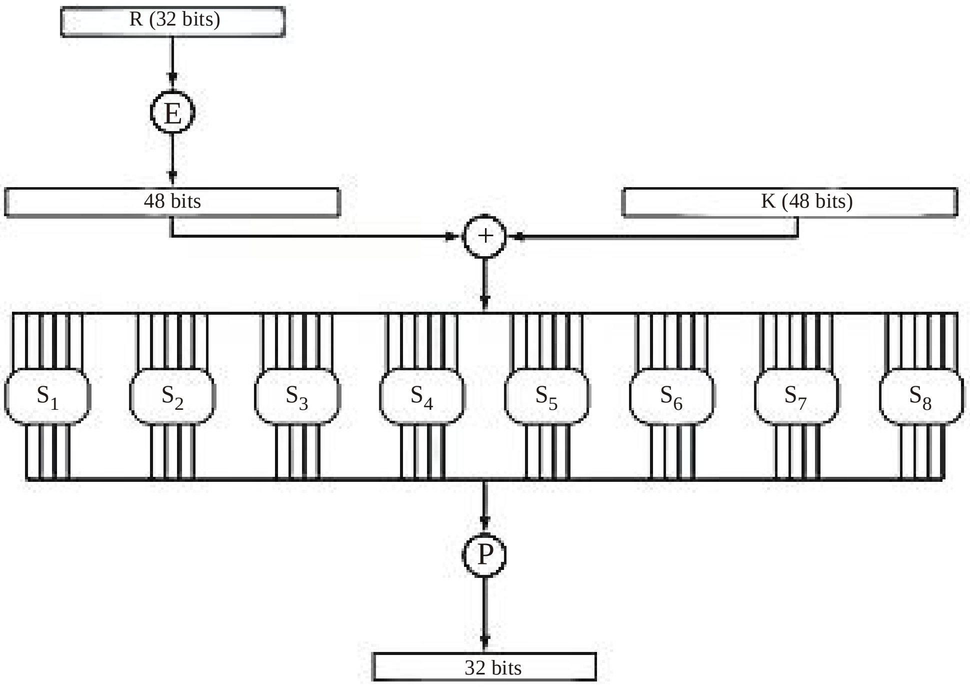


Fig. : Single Round of DES AlgorithmC

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

Key Generation

Returning to Figures 3.5 and 3.6, we see that a 64-bit key is used as input to the algorithm.The bits of the key are numbered from 1 through 64; every eighth bit is ignored, as indicated by the lack ofshading in Table 3.4a. The key is first subjected to a permutation governed by a table labelled Permuted Choice One (Table 3.4b). The resulting 56-bit key is then treated as two 28-bit quantities, labelled C0 and D0.At each round, Ci - 1 and Di - 1 are separately subjected to a circular left shift or (rotation) of 1 or 2 bits, asgoverned by Table 3.4d. These shifted values serve as input to the next round. They also serve as input to the part labelled Permuted Choice Two (Table 3.4c), which produces a 48-bit output that serves as input to the function F(Ri - 1, Ki).

DES Decryption

As with any Feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.

	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
S_1	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	13	0	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

(a) Input Key

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

(b) Permited Choice One (PC-I)

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

(c) Permuted Choice Two (PC-2)

14	17	11	24	1	5	3	28
15	6	21	10	23	19	12	4
26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40
51	45	33	48	44	49	39	56
34	53	46	42	50	36	29	52

(d) Schedule of Left Shits

Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Bits Rotated	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Table : DES Key Schedule Calculation

1.3.3 A DES Example

Q16. Illustrate DES with an example.

Ans :

We now work through an example and consider some of its implications. Although you are not expected to duplicate the example by hand, you will find it informative to study the hex patterns that occur from one step to the next.

For this example, the plaintext is a hexadecimal palindrome. The plaintext, key, and resulting ciphertext are asfollows:

Plaintext :	0 2 4 6 8 a c e e c a 8 6 4 2 0
Key :	0 f 1 5 7 1 c 9 4 7 d 9 e 8 5 9
Ciphertext :	d a 0 2 c e 3 a 8 9 e c a c 3 b

Results

Table 3.5 shows the progression of the algorithm. The first row shows the 32-bit values of the left and right halves of data after the initial permutation. The next 16 rows show the results after each round. Also shown is the value of the 48-bit subkey generated for each round. Note that Li = Ri - 1. The final row shows the left- and right-hand values after the inverse initial permutation. These two values com- bined form the ciphertext.

The Avalanche Effect

A desirable property of any encryption algorithm is that a small change in either the plaintext or the keyshould produce a significant change in the ciphertext. In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the ciphertext. This is referred to as the avalanche effect. If the change were small, this might provide a way to reduce the size of the plaintext or key space to be searched.

Using the example from Table 3.5, Table 3.6 shows the result when the fourth bit of the plaintext is changed,so that the plaintext is 12468aceeca86420. The second column of the table shows the intermediate 64-bitvalues at the end of each round for the two plaintexts. The third column shows the number of bits that differ between the two intermediate values. The table shows that, after just three rounds, 18 bits differ between thetwo blocks. On completion, the two ciphertexts differ in 32 bit positions.

Table :DES Example

Round	K_i	L_i	R_i
IP		5a005a00	3cf03c0f
1	1e030f03080d2930	3cf03c0f	bad22845
2	0a31293432242318	bad22845	99e9b723
3	23072318201d0c1d	99e9b723	0bae3b9e
4	05261d3824311a20	0bae3b9e	42415649
5	3325340136002c25	42415649	18b3fa41
6	123a2d0d04262a1c	18b3fa41	9616fe23
7	021f120b1c130611	9616fe23	67117cf2
8	1c10372a2832002b	67117cf2	c11bfc09
9	04292a380c341f03	c11bfc09	887fbc6c
10	2703212607280403	887fbc6c	600f7e8b
11	2826390c31261504	600f7e8b	f596506e
12	12071c241a0a0f08	f596506e	738538b8
13	300935393c0d100b	738538b8	c6a62c4e
14	311e09231321182a	c6a62c4e	56b0bd75
15	283d3e0227072528	56b0bd75	75e8fd8f
16	2921080b13143025	75e8fd8f	25896490
IP^{-1}		da02ce3a	89ecac3b

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Table : Avalanche Effect in DES : Change in Plaintext

Table shows a similar test using the original plaintext of with two keys that differ in only the fourth bit position: the original key, 0f1571c947d9e859, and the altered key, 1f1571c947d9e859. Again, the results show that about half of the bits in the ciphertext differ and that the avalanche effect is pronounced after just a fewrounds.

Round		δ
	02468aceeca86420 12468aceeca86420	1
1	3cf03c0fbad22845 3cf03c0fbad32845	1
2	bad2284599e9b723 bad3284539a9b7a3	5
3	99e9b7230bae3b9e 39a9b7a3171cb8b3	18
4	0bae3b9e42415649 171cb8b3ccaca55e	34
5	4241564918b3fa41 ccaca55ed16c3653	37
6	18b3fa419616fe23 d16c3653cf402c68	33
7	9616fe2367117cf2 cf402c682b2cefbc	32
8	67117cf2c11bfc09 2b2cefbc99f91153	33

Round		δ
9	c11bfc09887fbc6c 99f911532eed7d94	32
10	887fbc6c600f7e8b 2eed7d94d0f23094	34
11	600f7e8bf596506e d0f23094455da9c4	37
12	f596506e738538b8 455da9c47f6e3cf3	31
13	738538b8c6a62c4e 7f6e3cf34bc1a8d9	29
14	c6a62c4e56b0bd75 4bc1a8d91e07d409	33
15	56b0bd7575e8fd8f 1e07d4091ce2e6dc	31
16	75e8fd8f25896490 1ce2e6dc365e5f59	32
IP^{-1}	da02ce3a89ecac3b 057cde97d7683f2a	32

Table : Avalanche Effect in DES: Change in Key

1.3.4 The Strength of DES

Q17. Explain the strength of DES.

Ans :

Since its adoption as a federal standard, there have been lingering concerns about the level of security provided by DES. These concerns, by and large, fall into two areas: key size and the nature of the algorithm.

The Use of 56-Bit Keys

With a key length of 56 bits, there are 256 possible keys, which is approximately 7.2 * 1016 keys. Thus, on the face of it, a brute-force attack appears impractical. Assuming that, on average, half the key space has tobe searched, a single machine performing one DES encryption per microsecond would take more than athousand years to break the cipher.

It is important to note that there is more to a key-search attack than simply running through all possible keys. Unless known plaintext is provided, the analyst must be able to recognize plaintext as plaintext. If the message is just plain text in English, then the result pops out easily, although the task of recognizing English would have to beautomated. If the text message has been compressed before encryption, then recognition is more difficult.And if the message is some more general type of data, such as a numerical file, and this has been compressed, the problem becomes even more difficult to automate. Thus, to supplement the brute-force approach, some degree of knowledge about the expected plaintext is needed, and some means of automatically distinguishing plaintext from garble is also needed.

The EFF approach addresses this issue as well and introduces some automated techniques that would be effective in many contexts.

1.4 Block Cipher Operation

1.4.1 Double DES

Q18. Expalin about Double DES and attack on DES

Ans :

A method of data encryption, namely the use of DES twice in a row. Be P the plaintext message, k_1 and k_2 two DES keys, then the result of double DES is the ciphertext C = DES(DES(P,k_1),k_2).

As everybody proficient in cryptography knows, DES is not safe due to its fixed keylength of 56 bit, which is nowadays too little to be resilent against brute forceattacks. Unfortunately, double DES does not alleviate this problem - it's not much safer than a hypothetical DES with 57 bit keys, which is entirely too little. The reason why it doesn't offer the equivalent of 112 bit encryption that one might naively expect is its vulnerability to a so-called meet-in-the-middle attack.It works like this:

1. For every possible DES key k_i, compute M_i = DES(C,k_i) and store the tuple (M_i, k_i).
2. Loop again through all possible DES keys k_j and compute N_j = DES^{-1}(C,k_j). Check whether N_j is the same as one of your stored M_i. If you find a match, it's pretty likely likely that you've just found the two keys you're looking after!

Since the comparison in step two can be done in more or less constant time using hashing, you basically only have to go through all possible DES keys twice, resulting in the effective key lenght of 57 bit.

Attack on double DES:

i. The simplest form of multiple encryption has two encryption stages and two keys.

ii. Given a plaintext P and two encryption keys K1 and K2, cipher text C is generated as:

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

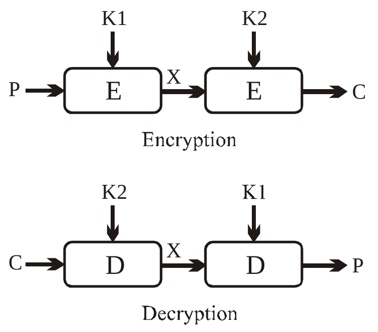
Decryption requires that the keys be applied in reverse order

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

iii. For DES, this scheme apparently involves a key length of 56x2 = 112 bits of resulting in a dramatic increase in cryptographic strength. But we need to examine the algorithm more closely.

iv. Meet in the middle attack: The algorithm known as a meet in the middle attack was first described in [DIFF 77]. It is based on the observation that if we have

C = E (K2, E (K1, P)) then X = E (K1, P) = D (K2, C)



Double Encryption

v. Given a known pair (P, C) the attack proceeds as follows.

vi. First, encrypt P for all 256256 possible values of K1 store these results in a table and then sort the table by the values of X.

vii. Next, decrypt C using all the 256256 possible values of K2. As each decryption is produced check the result against the table for a match.

viii. If a match occurs, then test the 2 resulting keys against a new known plain text – cipher text pair. If the 2 keys produce the correct cipher text, accept them as the correct keys.

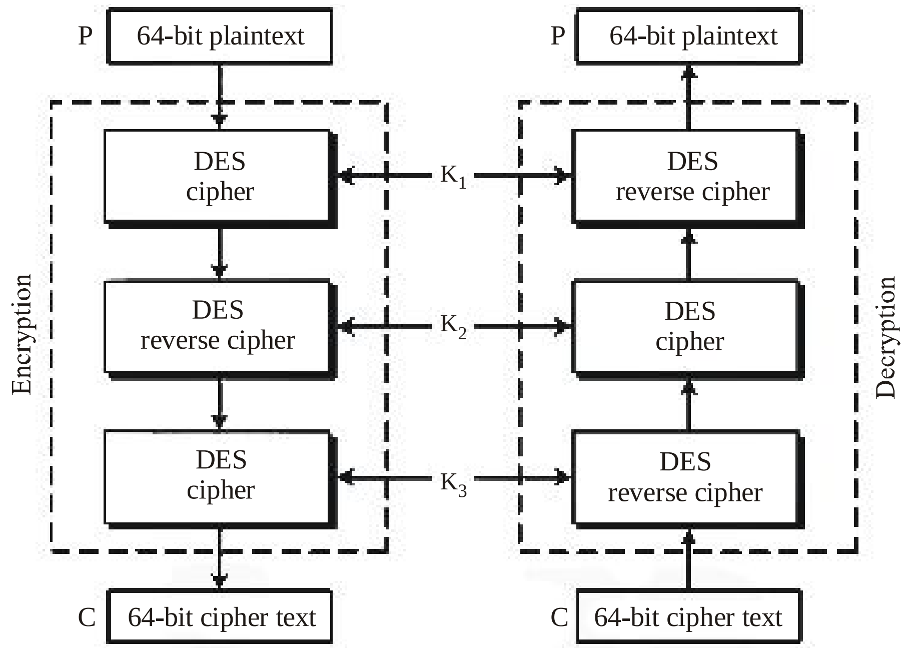
1.4.2 Triples DES

Q19. Explain triple DES with two keys.

Ans :

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 $3 \times 56 = 168$ bits.

The encryption scheme is illustrated as follows.



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.

1.4.3 Electronic Code Book

Q20. What is Electronic code Book ? Write a note on it.

Ans :

A block cipher takes a fixed-length block of text of length b bits and a key as input and produces a b-bitblock of ciphertext. If the amount of plaintext to be encrypted is greater than b bits, then the block cipher canstill be used by breaking the plaintext up into b-bit blocks. When multiple blocks of plaintext are encryptedusing the same key, a number of security issues arise. To apply a block cipher in a variety of applications, fivemodes of operation have been defined by NIST (SP 800-38A).

The five modes are intended to cover awide variety of applications of encryption for which a block cipher could be used. These modes are intendedfor use with any symmetric block cipher, including triple DES and AES. The modes are summarized inTable 6.1 and described in this and the following sections.

The simplest mode is the electronic codebook (ECB) mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key (Figure 6.3). The term codebook is usedbecause, for a given key, there is a unique ciphertext for every b-bit block of plaintext. Therefore, we can imagine a gigantic codebook in which there is an entry for every possible b-bit plaintext pattern showingits corresponding ciphertext.

For a message longer than b bits, the procedure is simply to break the message into b-bit blocks,padding the last block if necessary. Decryption is performed one block at a time, always using the samekey. In Figure 6.3, the plaintext (padded as necessary) consists of a sequence of b-bit blocks, P1, P2, Á ,PN; the

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 s bits at the 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 encryption output, and full blocks are used.	Stream-oriented transmission over noisy channel (e.g., satellite communication)
Counter (CTR)	Each block of plaintext is XORed with an encrypted courier. The counter is incremented for cash subsequent block.	General-purpose block-oriented transmission. Useful for high-speed requirements.

Table : Block Cipher Modes of Operation

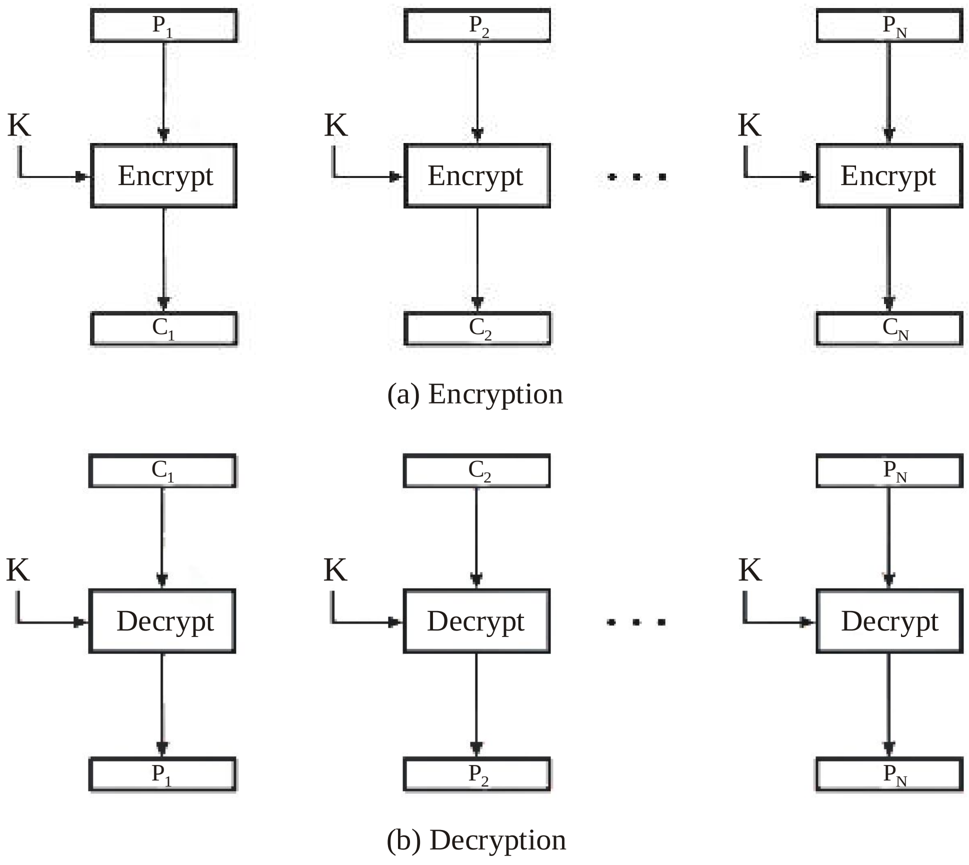


Fig. : Electronic Codebook (ECB) Mode

corresponding sequence of ciphertext blocks is $C1$, $C2$, ... , CN. We can define ECB mode as follows.

ECB	$C_j = E(K, P_j)$ $\quad j = 1, \ldots, N$	$P_j = D(K, C_j)$ $\quad j = 1, \ldots, N$

The ECB method is ideal for a short amount of data, such as an encryption key. Thus, if you want to transmita DES or AES key securely, ECB is the appropriate mode to use.

The most significant characteristic of ECB is that if the same b-bit block of plaintext appears more than once in the message, it always produces the same ciphertext.

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

1.4.4 Cipher Block Chaining Mode

Q21. What is cipher block chaining mode? Explain.

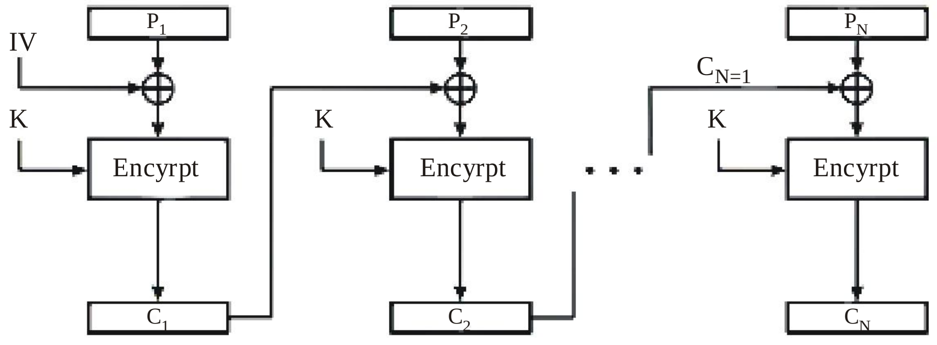
Ans :

To overcome the security deficiencies of ECB, we would like a technique in which the same plaintext block,if repeated, produces different ciphertext blocks. A simple way to satisfy this requirement is the cipher blockchaining (CBC) mode (Figure 6.4). In this scheme, the input to the encryption algorithm is the XOR of thecurrent plain- text block and the preceding ciphertext block; the same key is used for each block. In effect, wehave chained together the processing of the sequence of plaintext blocks. The input to the encryption functionfor each plaintext block bears no fixed relation- ship to the plaintext block. Therefore, repeating patterns of bbits are not exposed. As with the ECB mode, the CBC mode requires that the last block be padded to a full bbits if it is a partial block.

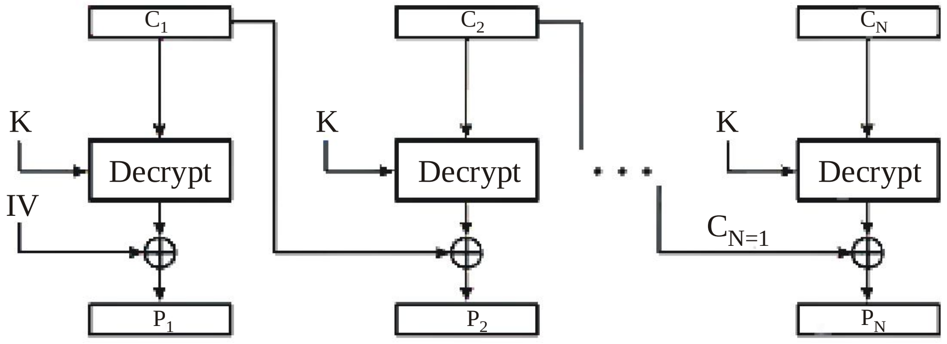
For decryption, each cipher block is passed through the decryption algorithm. The result is XORed with the preceding ciphertext block to produce the plaintext block. To see that this works, we can writeCj = E(K, [Cj - 1 Í$ Pj])

Then

$$D(K,C_j) = D\ (K, E(K, [C_{j-1} \oplus P_j]))$$
$$D(K,C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$
$$C_{j-1} \oplus D(K,C_j) = C_{j-1} \oplus C_{j-1} \oplus P_j = P_j$$



(a) Encryption



(b) Decryption

To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext. On decryption, the IV is XORed with the output of the decryption algorithm to recover the first block of plaintext. The IV is a data block that is that same size as the cipher block. We can define CBC mode as

CBC	$C_1 = \mathrm{E}(K, [P_1 \oplus \mathrm{IV}])$ $C_j = \mathrm{E}(K, [P_j \oplus C_{j-1}])\ j = 2, \ldots, N$	$P_1 = \mathrm{D}(K, C_1) \oplus \mathrm{IV}$ $P_j = \mathrm{D}(K, C_j) \oplus C_{j-1}\ \ j = 2, \ldots, N$

The IV must be known to both the sender and receiver but be unpredictable by a third party. In particular, forany given plaintext, it must not be possible to predict the IV that will be associated to the plaintext in advance of the generation of the IV. To see this, consider

$$C1 = \mathrm{E}(K, [\mathrm{IV} \otimes P1])$$

$$P1 = \mathrm{IV} \otimes \mathrm{D}(K, C1)$$

Now use the notation that $X[i]$ denotes the ith bit of the b-bit quantity X. Then

$$P1[i] = \mathrm{IV}[i] \otimes \mathrm{D}(K, C1)[i]$$

Then, using the properties of XOR, we can state

$$P1[i]' = \mathrm{IV}[i]' \otimes \mathrm{D}(K, C1)[i]$$

where the prime notation denotes bit complementation. This means that if an opponent can predictablychange bits in IV, the corresponding bits of the received value of $P1$ can be changed.

1.4.5 Cipher Feedback Mode

Q22. Explain about Cipher Feedback Mode.

Ans :

A stream cipher eliminates the need to pad amessage to be an integral number of blocks. It also can operate in real time. Thus, if a character stream isbeing transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher.

One desirable property of a stream cipher is that the ciphertext be of the same length as the plaintext. Thus, if8-bit characters are being transmitted, each character should be encrypted to produce a ciphertext output of 8bits. If more than 8 bits are produced, transmission capacity is wasted.

Figure 6.5 depicts the CFB scheme. In the figure, it is assumed that the unit of transmission is s bits; acommon value is s = 8. As with CBC, the units of plaintext are chained together, so that the ciphertext of any plaintext unit is a function of all the preceding plaintext. In this case, rather than blocks of b bits, the plaintext is divided into segments of s bits.

First, consider encryption. The input to the encryption function is a b-bit shift register that is initially set tosome initialization vector (IV). The leftmost (most significant) s bits of the output of the encryption function are XORed with the first segment of plaintext P1 to produce the first unit of ciphertext C1, whichis then transmitted. In addition, the contents of the shift register are shifted left by s bits, and C1 is placed in the rightmost (least significant) s bits of the shift register. This process continues until all plaintext unitshave been encrypted.

For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit. Note that it is the encryption function that is used, not the decryption function. This is easily explained. Let MSBs(X) be defined as the most significant s bits of X. Then

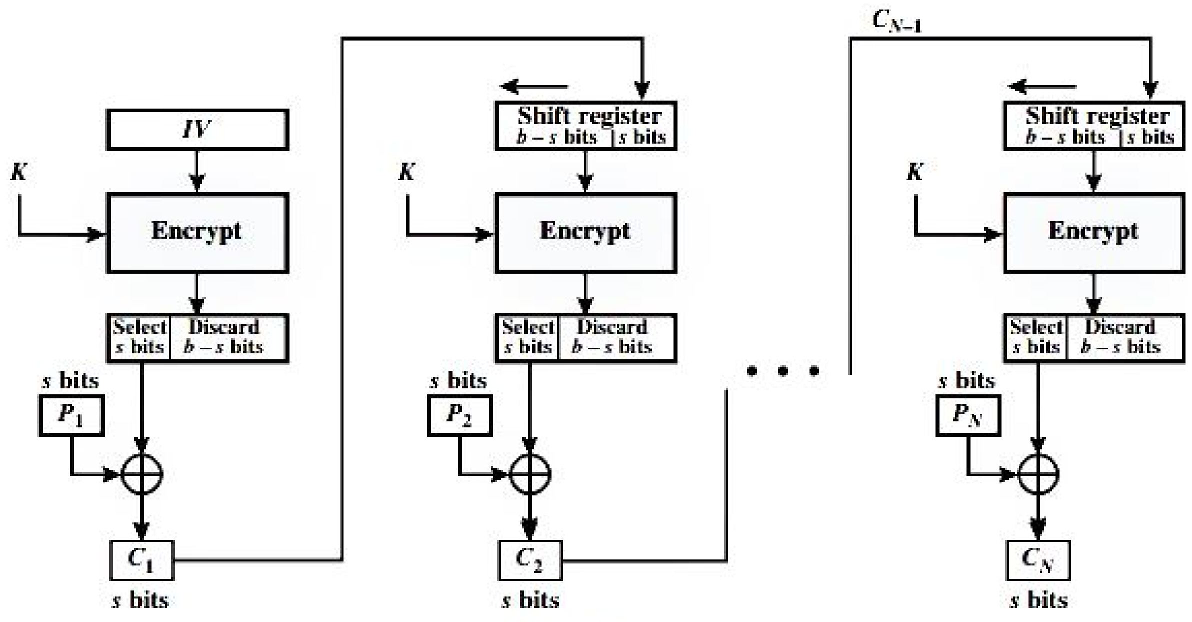
C1 = P1 Í$ MSBs[E(*K*, IV)]

Therefore, by rearranging terms:

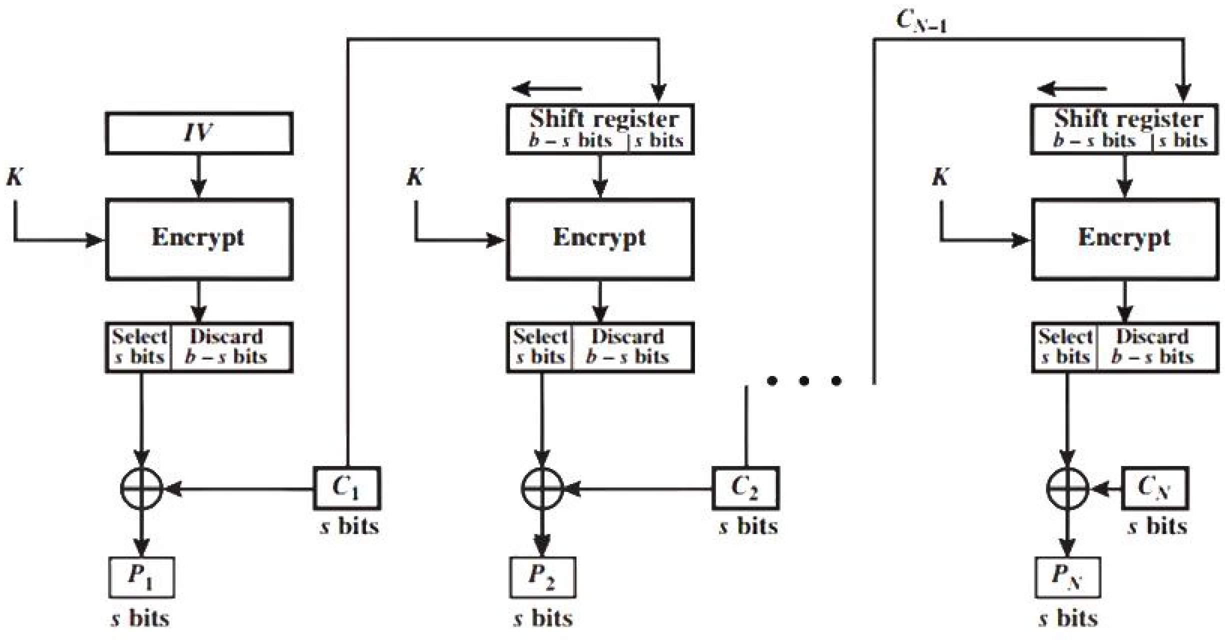
*P*1 = *C*1 Í$ MSBs[E(*K*, IV)]

The same reasoning holds for subsequent steps in the process.

We can define CFB mode as follows.



(a) Encryption



(b) Decryption

s-bit Cipher Feedback (CFB) Mode

CFB	$I_1 = IV$ $I_j = \text{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \ldots, N$ $O_j = \text{E}(K, I_j) \quad j = 1, \ldots, N$ $C_j = P_j \oplus \text{MSB}_s(O_j) \quad j = 1, \ldots, N$	$I_1 = IV$ $I_j = \text{LSB}_{b-s}(I_{j-1}) \parallel C_{j-1} \quad j = 2, \ldots, N$ $O_j = \text{E}(K, I_j) \quad j = 1, \ldots, N$ $P_j = C_j \oplus \text{MSB}_s(O_j) \quad j = 1, \ldots, N$

Although CFB can be viewed as a stream cipher, it does not conform to the typical construction of a streamcipher. In a typical stream cipher, the cipher takes as input some initial value and a key and generates astream of bits, which is then XORed with the plaintext bits (see Figure 3.1). In the case of CFB, the stream ofbits that is XORed with the plaintext also depends on the plaintext.

1.4.6 Output Feedback Mode

Q23. What is Output Feedback Mode? Explain.

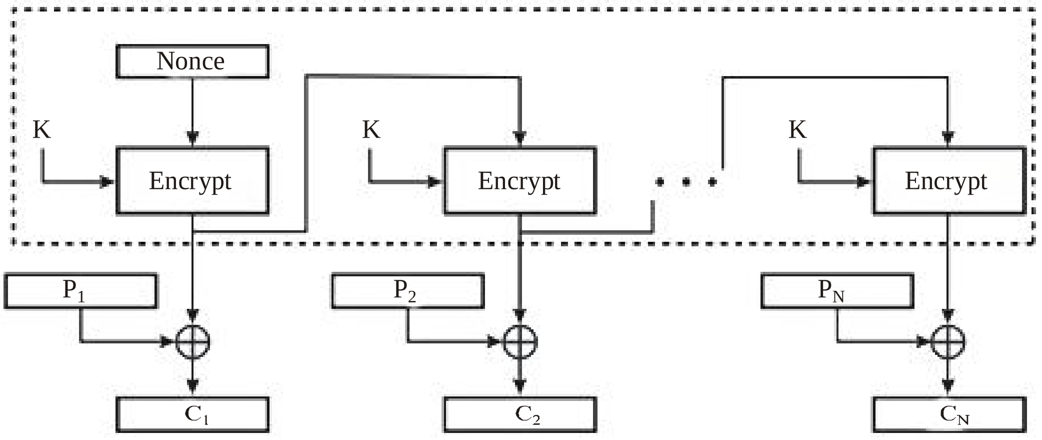
Ans :

The output feedback (OFB) mode is similar in structure to that of CFB. As can be seen in Figure 6.6, it isthe output of the encryption function that is fed back to the shift register in OFB, whereas in CFB, theciphertext unit is fed back to the shift register. The other difference is that the OFB mode operates on fullblocks of plaintext and ciphertext, not on an s-bit subset. Encryption can be expressed as

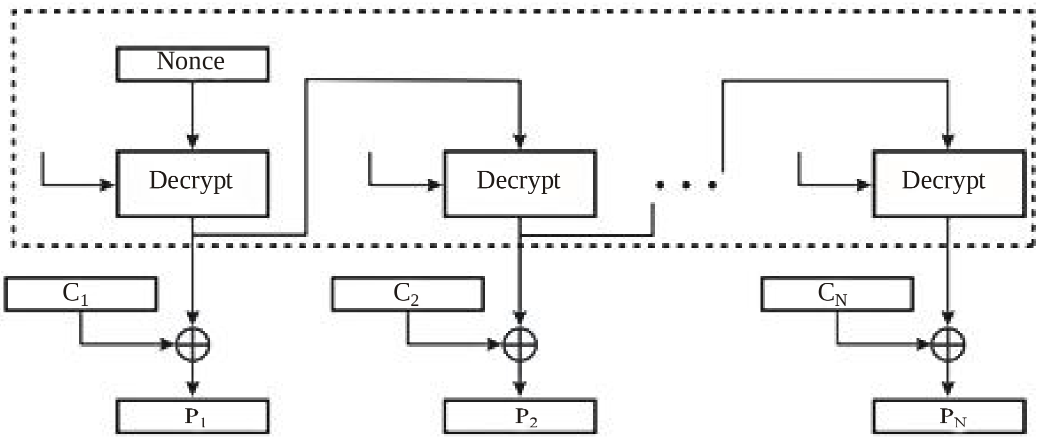
$$Cj = Pj \otimes \text{E}(K, [C_{j-1} \otimes P_{j-1}])$$

By rearranging terms, we can demonstrate that decryption works.

$$Pj = Cj \otimes \text{E}(K, [C_{j-1} \otimes P_{j-1}])$$



(a) Encryption



(b) Decryption

Fig. : Cipher Block chaining (CFB) Mode

Let the size of a block be b. If the last block of plaintext contains u bits, with u 6 b, themost significant u bits of the last output block ON are used for the XOR operation; the remaining b -u bits ofthe last output block are discarded.

As with CBC and CFB, the OFB mode requires an initialization vector. In the case of OFB, the IV must be anonce; that is, the IV must be unique to each execution of the encryption operation. The reason for this isthat the sequence of encryption output blocks, Oi, depends only on the key and the IV and does not dependon the plaintext. Therefore, for a given key and IV, the stream of output bits used to XOR with the stream of plaintext bits is fixed. If two different messages had an identical block of plaintext in the identical position,then an attacker would be able to determine that portion of the Oi stream.

One advantage of the OFB method is that bit errors in transmission do not propagate.

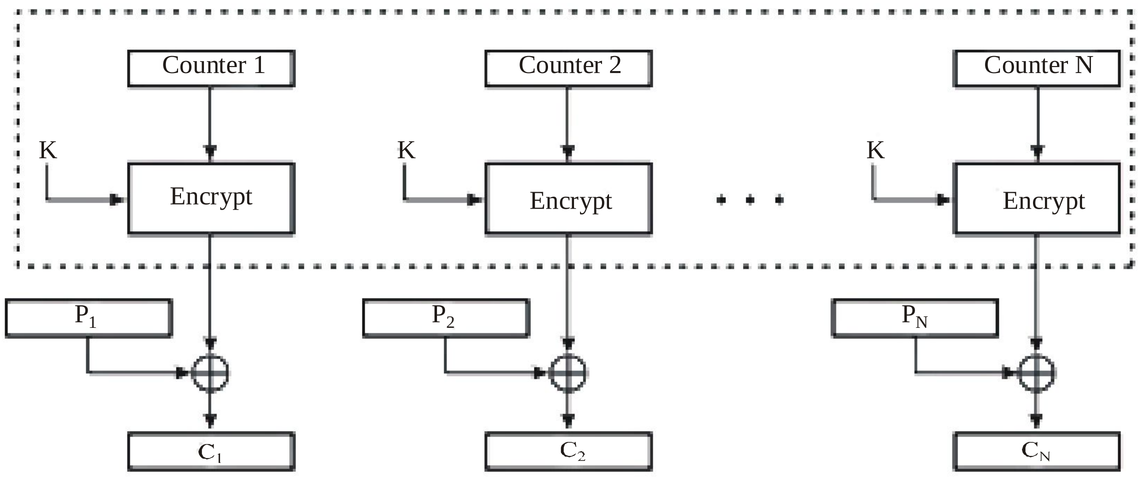
OFB has the structure of a typical stream cipher, because the cipher generates a stream of bits as a functionof an initial value and a key, and that stream of bits is XORed with the plaintext bits (see Figure 3.1). Thegenerated stream that is XORed with the plaintext is itself independent of the plaintext.

1.4.7 Counter Mode

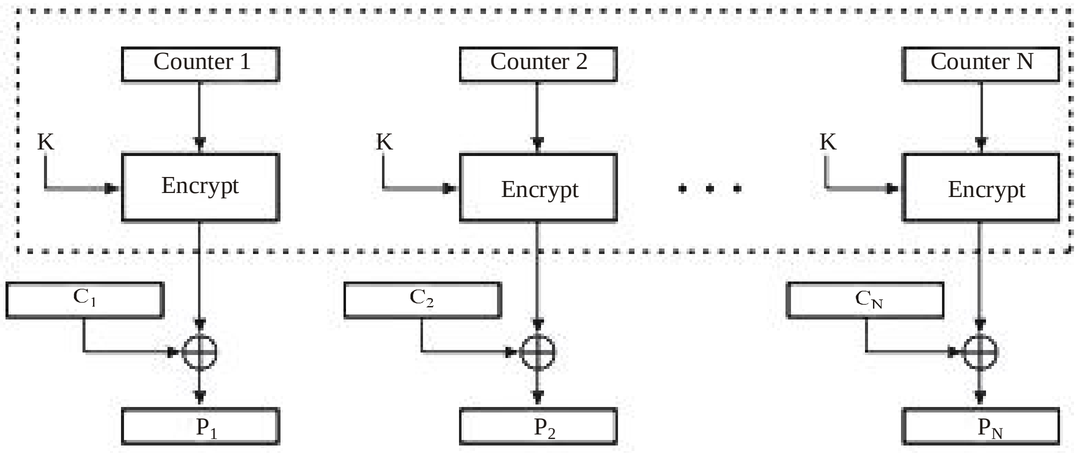
Q24. What is counter Mode ? Explain.

Ans :

Figure depicts the CTR mode. A counter equal to the plaintext block size is used. The only requirement stated is that the counter value must be



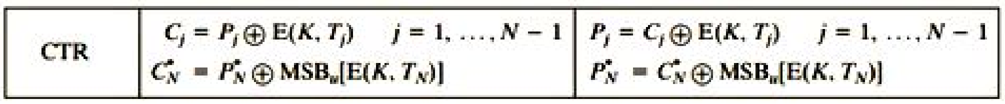
(a) Encryption



(b) Decryption

different for each plaintext block that is encrypted. Typically, the counter is initialized to some value and thenincremented by 1 for each subsequent block (modulo 2b, where b is the block size). For encryption, the counteris encrypted and then XORed with the plaintext block to produce the ciphertext block; there is no chaining. Fordencryption, the same sequence of counter values is used, with each encrypted counter XORed with a ciphertextblock to recover the corresponding plaintext block. Thus, the initial counter value must be made available fordecryption. Given a sequence of counters T1, T2, ...,TN, we can define CTR mode as follows.

For the last plaintext block, which may be a partial block of u bits, the most significant u bits of the last outputblock are used for the XOR operation; the remaining b -u bits are discarded. Unlike the ECB, CBC, and CFBmodes, we do not need to use padding because of the structure of the CTR mode.



CTR	$C_j = P_j \oplus E(K, T_j) \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus MSB_u[E(K, T_N)]$	$P_j = C_j \oplus E(K, T_j) \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus MSB_u[E(K, T_N)]$

As with the OFB mode, the initial counter value must be a nonce; that is, T1 must be different for all of themessages encrypted using the same key. Further, all Ti values across all messages must be unique. If, contraryto this requirement, a counter value is used multiple times, then the confidentiality of all of the plaintext blocks corresponding to that counter value may be compromised. In particular, if any plain- text block that isencrypted using a given counter value is known, then the output of the encryption function can be determined easily from the associated ciphertext block. This output allows any other plaintext blocks that are encrypted using the same counter value to be easily recovered from their associated ciphertext blocks.

One way to ensure the uniqueness of counter values is to continue to increment the counter value by 1 across messages. That is, the first counter value of the each message is one more than the last counter valueof the preceding message.

Advantages of CTR mode.

1. **Hardware efficiency**

   For the chaining modes, the algorithm must complete the computation on one block before beginning on the next block. This limits the maximum throughput of the algorithm to the reciprocal of the time for one execution of block encryption ordecryption. In CTR mode, the throughput is only limited by the amount of parallelism that is achieved.

2. **Software efficiency**

   Similarly, because of the opportunities for parallel execution in CTR mode,processors that support parallel features, such as aggressive pipelining, multiple instruction dispatch per clockcycle, a large number of registers, and SIMD instructions, can be effectively utilized.

3. **Preprocessing**

   The execution of the underlying encryption algorithm does not depend on input ofthe plaintext or ciphertext. Therefore, if sufficient memory is available and security is maintained, preprocessing can be used to prepare the output of the encryption boxes.

4. **Random access:** The ith block of plaintext or ciphertext can be processed in random-access fashion. With the chaining modes, block Ci cannot be comuted until the i – 1 prior block are computed.

5. **Provable security:** It can be shown that CTR is at least as secure as the other modes discussed in this section.

6 **Simplicity:** Unlike ECB and CBC modes, CTR mode requires only the implementation of theencryption algorithm and not the decryption algorithm. This matters most when the decryption algorithm differs substantially from the encryption algorithm, as it does for AES.

UNIT II

Advanced Encryption Standard (AES): The Origins AES, AES Structure, AES Round Functions, AES Key Expansion, an AES Example AES Implementation.Pseudorandom Number Generation and Stream Ciphers: Principles of Pseudorandom Number Generation, Pseudorandom Number Generators,Pseudorandom Number Generation using BlockCipher, StreamCiphers-RC4. Public-Key Cryptography and RSA: Principles of Public-Key Cryptosystems, the RSA Algorithm.Key Management and Distribution: Symmetric Key Distribution Using Symmetric Encryption and Asymmetric Encryption, Distribution of Public Keys, X.509 Certificates, Diffie- Hellman Key Exchange.

2.1 Advanced Encryption Standard

2.1.1 The Origions of AES

Q1. Explain about the features of AES.

Ans :

In 1997, National Institute of Standards and Technology NIST issued a call for proposals for a new Advanced Encryption Standard (AES), which should have security strength equal to or better than 3DES, and significantly improved efficiency. In addition, NIST also specified that AES must be a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192, and 256 bits.

AES Features

The selection process for this new symmetric key algorithm was fully open to public scrutiny and comment; this ensured a thorough, transparent analysis of the designs submitted.

NIST specified the new advanced encryption standard algorithm must be a block cipher capable of handling 128 bit blocks, using keys sized at 128, 192, and 256 bits; other criteria for being chosen as the next advanced encryption standard algorithm included:

- **Security:** Competing algorithms were to be judged on their ability to resist attack, as compared to other submitted ciphers, though security strength was to be considered the most important factor in the competition.
- **Cost:** Intended to be released under a global, non exclusive and royalty-free basis, the candidate algorithms were to be evaluated on computational and memory efficiency.
- **Implementation:** Algorithm and implementation characteristics to be evaluated included the flexibility of the algorithm; suitability of the algorithm to be implemented in hardware or software; and overall, relative simplicity of implementation.

AES Structure

Q2. Explain about the structure of AES.

Ans :

Figure 5.3 shows the AES cipher in more detail, indicating the sequence of transfor- mations in each round and showing the corresponding decryption function.

1. One note worthy feature of this structure is that it is not a Feistel structure. AES instead processes the entire data block as a single matrix during each round using substitutions and permutation.
2. The key that is provided as input is expanded into an array of forty-four 32-bit words, w[i]. Four distinctwords (128 bits) serve as a round key for each round; these are indicated in Figure 5.3.
3. Four different stages are used, one of permutation and three of substitution:
   - Substitute bytes: Uses an S-box to perform a byte-by-byte substitution of the block
   - Shift Rows: A simple permutation
   - Mix Columns: A substitution that makes use of arithmetic over GF(28)
   - AddRoundKey: A simple bitwise XOR of the current block with a portion of the expanded key

4. The structure is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round ofthree stages. Figure 5.4 depicts the structure of a full encryption round.
5. Only the AddRoundKey stage makes use of the key. For this reason, the cipher begins and ends with an AddRoundKey stage. Any other stage, applied at the beginning or end, is reversible without knowledge of thekey and so would add no security.
6. The AddRoundKey stage is, in effect, a form of Vernam cipher and by itself would not be formidable. The other three stages together provide confusion, diffusion, and non linearity.
7. Each stage is easily reversible. For the Substitute Byte, ShiftRows, and MixColumns stages, aninverse function is used in the decryption algorithm. For the AddRoundKey stage, the inverse is achieved by XO Ring the same round key to the block, using the result that A $\otimes$ B $\otimes$ B = A.
8. As with most block ciphers, the decryption algorithm makes use of the expanded key in reverse order. However, the decryption algorithm is not identical to the encryption algorithm. This is a consequence of the particular structure of AES.
9. Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext.
10. The final round of both encryption and decryption consists of only three stages. Again, this is a consequence of the particular structure of AES and is required to make the cipher reversible.

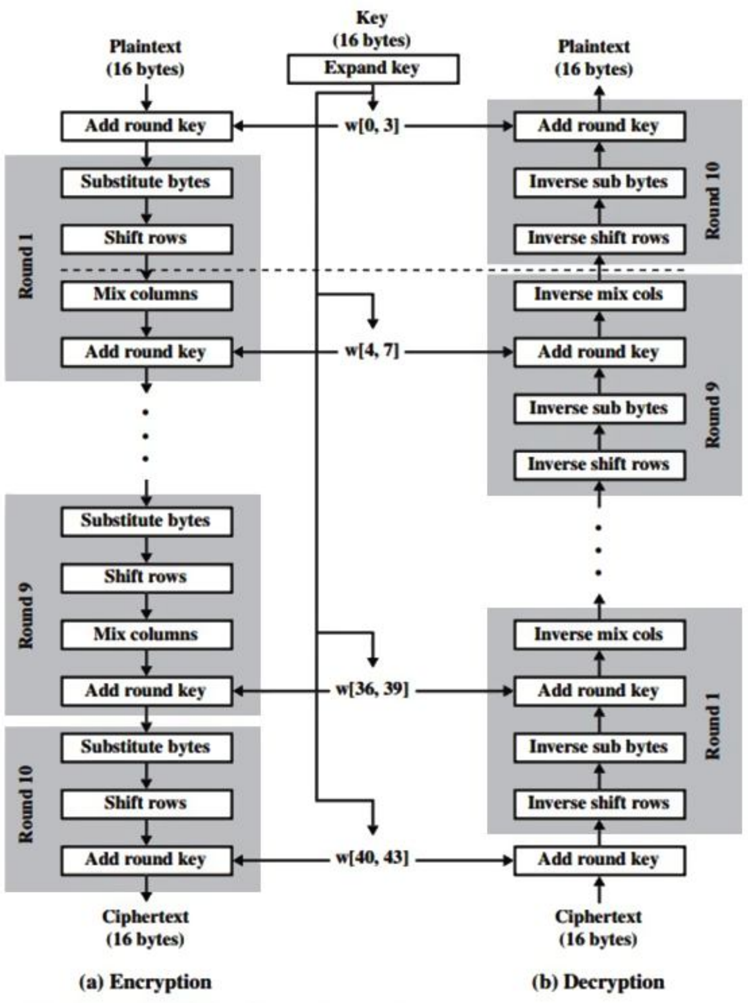


Figure : AES Encryption and Decryption

Q3. Explain AES Round function.

Ans :

Round Keys

The cipher key used for encryption is 128 bits long. The cipher key is already the result of many hashing and cryptographic transformations and, by the time it arrives at the AES block encryption, it is far removed from the secret master key held by the authentication server. Now, finally, it is used to generate a set of eleven 128-bit round keys that will be combined with the data during encryption.

Although there are ten rounds, eleven keys are needed because one extra key is added to the initial state array before the rounds start. The best way to view these keys is an array of eleven 16-byte values, each made up of four 32-bit words, as shown in Table A.6.

To start with, the first round key Rkey0 is simply initialized to the value of the cipher key (that is the secret key delivered through the key hierarchy). Each of the remaining ten keys is derived from this as follows.

	32 bits	32 bits	32 bits
Rkey0	W0	W1	W2
Rkey1	W0	W1	W2
Rkey2	W0	W1	W2
Rkey3	W0	W1	W2
Rkey4	W0	W1	W2
Rkey5	W0	W1	W2
Rkey6	W0	W1	W2
Rkey7	W0	W1	W2
Rkey8	W0	W1	W2
Rkey9	W0	W1	W2
Rkey10	W0	W1	W2

Table : Round Key Array

For each of the round keys Rkey1 to Rkey10, words W1, W2, W3 are computed as the sum[1] of the corresponding word in the previous round key and the preceding word in the current round key. For example, using XOR for addition:

[1] Using finite field arithmetic.

Rkey5: W1 = Rkey4:W1 XOR Rkey5:W0,

Rkey8: W3 = Rkey7:W3 XOR Rkey8:W2 and so on.

The rule for the value of W0 is a little more complicated to describe, although still simple to compute. For each round key Rkey1 to Rkey10, the value of W0 is the sum of three 32-bit values:

- The value of W0 from the previous round key
- The value of W3 from the previous round key, rotated right by 8 bits
- A special value from a table called Rcon

Thus, we write:

Rkeyi: W0 = Rkey(i-1):W0 XOR (Rkey(i-1):W3 >>> 8) XOR Rcon[i]

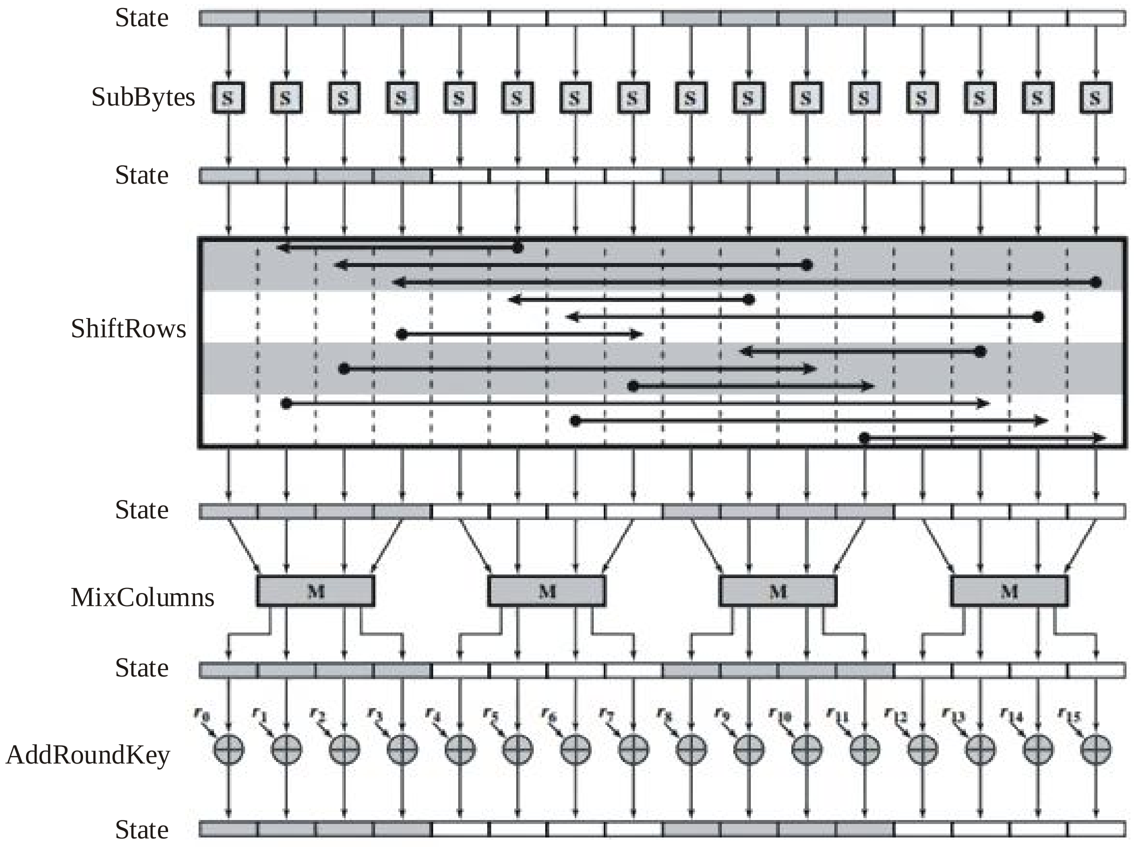

State
SubBytes
S
State
ShiftRows
State
MixColumns
M
State
AddRoundKey
r_0 r_1 r_2 r_3 r_4 r_5 r_6 r_7 r_8 r_9 r_{10} r_{11} r_{12} r_{13} r_{14} r_{15}
State

Fig. : AES Encryption Round

2.1.2 AES Key Expansion

Q4. Explain AES Key Expansion Algorithm.

Ans :

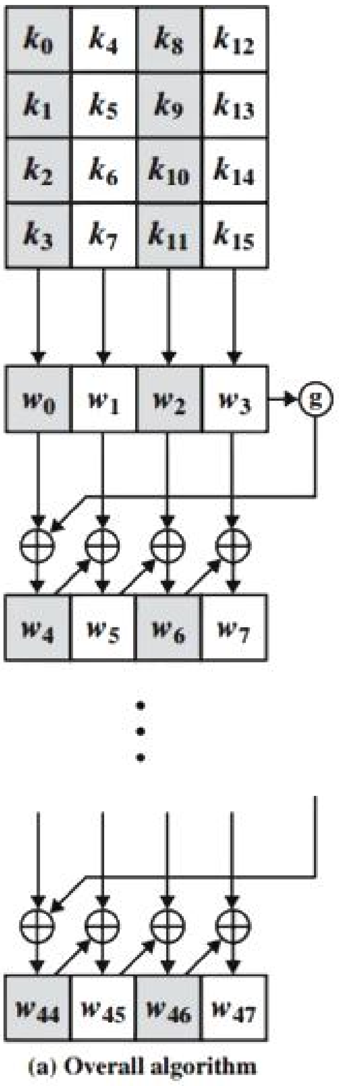
Key Expansion Algorithm

The AES key expansion algorithm takes as input a four-word (16-byte) key and produces a linear array of 44words (176 bytes). This is sufficient to provide a four-word round key for the initial AddRoundKey stage andeach of the 10 rounds of the cipher. The pseudocode on the next page describes the expansion.

The key is copied into the first four words of the expanded key. The remain- der of the expanded key isfilled in four words at a time. Each added word w[i] depends on the immediately preceding word, w[i - 1], and the word four positions back, w[i - 4]. In three out of four cases, a simple XOR is used. For a wordwhose position in the w array is a multiple of 4, a more complex function is used. Figure 5.9 illustrates thegeneration of the expanded key, using the symbol g to represent that complex function. The function consists of the following subfunctions.

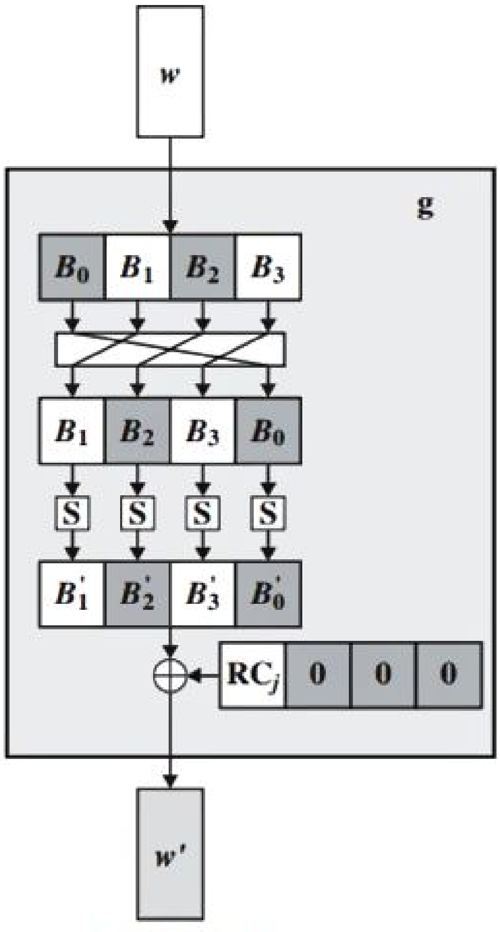
```
KeyExpansion (byte key[16], word w[44])
{
    word temp
    for (i = 0; i < 4; i++)   w[i] = (key[4*i], key[4*i+1],
                                    key[4*i+2],
                                    key[4*i+3]);

    for (i = 4; i < 44; i++)
    {
     temp = w[i - 1];
     if (i mod 4 = 0)    temp = SubWord (RotWord (temp))
                                ⊕ Rcon[i/4];
     w[i] = w[i-4]  ⊕  temp
    }
}
```



k_0 k_4 k_8 k_{12}
k_1 k_5 k_9 k_{13}
k_2 k_6 k_{10} k_{14}
k_3 k_7 k_{11} k_{15}
w_0 w_1 w_2 w_3
g
w_4 w_5 w_6 w_7
w_{44} w_{45} w_{46} w_{47}

(a) Overall algorithm



(b) Function g

Fig. : AES Key Expansion

1. RotWord performs a one-byte circular left shift on a word. This means that an input word [B0, B1, B2, B3] is transformed into [B1, B2, B3, B0].
2. SubWord performs a byte substitution on each byte of its input word, using the S-box (Table 5.2a).
3. The result of steps 1 and 2 is XORed with a round constant, Rcon[j].

The round constant is a word in which the three rightmost bytes are always 0. Thus, the effect of an XOR of aword with Rcon is to only perform an XOR on the left- most byte of the word. The round constant is different foreach round and is defined as Rcon[j] = (RC[j], 0, 0, 0), with RC[1] = 1, RC[j] = 2 RC[j -1] and with multiplication defined over the field GF(28). The values of RC[j] in hexadecimal are

J	1	2	3	4	5	6	7	8	9	10
RC[j]	01	02	04	08	10	20	40	80	1B	36

For example, suppose that the round key for round 8 is

EA D2 73 21 B5 8D BA D2 31 2B F5 60 7F 8D 29 2F

Then the first 4 bytes (first column) of the round key for round 9 are calculated as follows:

I(decimal)	Temp	After RotWord	After SubWord	Rcon (9)	After XOR With Rcon	W[i-4]	W[i] = temp ⊕ W[o-4]
36	7FSD292F	SD292F7F	5DA515D2	1B000000	46A515D2	EAD27321	AC7766F3

Rationale

The Rijndael developers designed the expansion key algorithm to be resistant to known cryptanalytic attacks. The inclusion of a round-dependent round constant eliminates the symmetry, or similarity, between the ways in which round keys are generated in different rounds. The specific criteria that were used are[DAEM99]

- Knowledge of a part of the cipher key or round key does not enable calculation of many otherround-key bits.
- An invertible transformation [i.e., knowledge of any Nk consecutive words of the expanded key enables regeneration the entire expanded key (Nk = key size in words)].
- Speed on a wide range of processors.
- Usage of round constants to eliminate symmetries.
- Diffusion of cipher key differences into the round keys; that is, each key bit affects many round keybits.

- Enough nonlinearity to prohibit the full determination of round key differences from cipher keydifferences only.
- Simplicity of description

2.1.3 An AES Example

Q5. Illustrate AES with an example.

Ans :

For this example, the plaintext is a hexadecimal palindrome. The plaintext, key, and resulting ciphertext are

Plaintext	:	0123456789abcdeffedcba9876543210
Key	:	0f1571c947d9e8590cb7add6af7f6798
Ciphertext	:	ff0b844a0853bf7c6934ab4364148fb9

Results

Table 5.3 shows the expansion of the 16-byte key into 10 round keys. As previously explained, this process is performed word by word, with each four-byte word occupying one column of the word round-key matrix. The left-hand column shows the four round-key words generated for each round. The right-hand column shows the steps

Table : Key Expansion for AES Example

Key Words	Auxiliary Function
w0 = 0f 15 71 c9	RotWord(w3) = 7f 67 98 af = x1
w1 = 47 d9 e8 59	SubWord(x1) = d2 85 46 79 = y1
w2 = 0c b7 ad	Rcon(1) = 01 00 00 00
w3 = af 7f 67 98	y1 ⊕ Rcon(1) = d3 85 46 79 = z1
w4 = w0 ⊕ z1 = dc 90 37 b0	RotWord(w7) = 81 15 a7 38 = x2
w5 = w4 ⊕ w1 = 9b 49 df e9	SubWord(x4) = 0c 59 5c 07 = y2
w6 = w5 ⊕ w2 = 97 fe 72 3f	Rcon(2) = 02 00 00 00
w7 = w6 ⊕ w3 = 38 81 15 a7	y2 ⊕ Rcon(2) = 0e 59 5c 07 = z2
w8 = w4 ⊕ z2 = d2 c9 6b b7	RotWord(w11) = ff d3 c6 e6 = x3
w9 = w8 ⊕ w5 = 49 80 b4 5e	SubWord(x2) = 16 66 b4 83 = y3
w10 = w9 ⊕ w6 = de 7e c6 61	Rcon(3) = 04 00 00 00
w11 = w10 ⊕ w7 = e6 ff d3 c6	y3 ⊕ Rcon(3) = 12 66 b4 8e = z3
w12 = w8 ⊕ z3 = c0 af df 39	RotWord(w15) = ae 7e c0 b1 = x4
w13 = w12 ⊕ w9 = 89 2f 6b 67	SubWord(x3) = e4 f3 ba c8 = y4
w14 = w13 ⊕ w10 = 57 51 ad 06	Rcon(4) = 08 00 00 00
w15 = w14 ⊕ w11 = b1 ae 7e c0	y4 ⊕ Rcon(4) = ec f3 ba c8 = 4

Key Words	Auxiliary Function
w16 = w12 ⊕ z4 = 2c 5c 65 f1	RotWord(w19) = 8c dd 50 43 = x5
w17 = w16 ⊕ w13 = a5 73 0e 96	SubWord(x4) = 64 c1 53 1a = y5
w18 = w17 ⊕ w14 = f2 22 a3 90	Rcon(5) = 10 00 00 00
w19 = w18 ⊕ w15 = 43 8c dd 50	y5 ⊕ Rcon(5) = 74 c1 53 1a = z5
w20 = w16 ⊕ z5 = 58 9d 36 eb	RotWord (w23) = 40 46 bd 4c = x6
w21 = w20 ⊕ w17 = fd ee 38 7d	SubWord (x5) = 09 5a 7a 29 = y6
w22 = w21 ⊕ w18 = 0f cc 9b ed	Rcon(6) = 20 00 00 00
w23 = w22 ⊕ w19 = 4c 40 46 bd	y6 ⊕ Rcon(6) = 29 5a 7a 29 = z6
w24 = w20 ⊕ z6 = 71 c7 4c c2	RotWord (w27) = a5 a9 ef cf = x7
w25 = w24 ⊕ w21 = 8c 29 74 bf	SubWord (x6) = 06 d3 bf 8a = y7
w26 = w25 ⊕ w22 = 83 e5 ef 52	Rcon (7) = 40 00 00 00
w27 = w26 ⊕ w23 = cf a5 a9 ef	y7 ⊕ Rcon(7) = 46 d3 df 8a = z7
w28 = w24 ⊕ z7 = 37 14 93 48	RotWord (w31) = 7d a1 4a f7 = x8
w29 = w28 ⊕ w25 = bb 3d e7 f7	SubWord (x7) = ff 32 d6 68 = y8
w30 = w29 ⊕ w26 = 38 d8 08 a5	Rcon (8) = 80 00 00 00
w31 = w30 ⊕ w27 = f7 7d a1 4a	y8 ⊕ Rcon(8) = 7f 32 d6 68 = z8
w32 = w28 ⊕ z8 = 48 26 45 20	RotWord (w35) = be 0b 38 3c = x9
w33 = w32 ⊕ w29 = f3 1b a2 d7	SubWord (x8) = ae 2b 07 eb = y9
w34 = w33 ⊕ w30 = cb c3 aa 72	Rcon (9) = 1B 00 00 00
w35 = w34 ⊕ w32 = 3c be 0b 3	y9 ⊕ Rcon (9) = b5 2b 07 eb = z9
w36 = w32 ⊕ z9 = fd 0d 42 cb	RotWord (w39) = 6b 41 56 f9 = x10
w37 = w36 ⊕ w33 = 0e 16 e0 1c	SubWord (x9) = 7f 83 b1 99 = y10
w38 = w37 ⊕ w34 = c5 d5 4a 6e	Rcon (10) = 36 00 00 00
w39 = w38 ⊕ w35 = f9 6b 41 56	y10 ⊕ Rcon (10) = 49 83 b1 99 = z10
w40 = w36 ⊕ z10 = b4 8e f3 52	
w41 = w40 ⊕ w37 = ba 98 13 4e	
w42 = w41 ⊕ w38 = 7f 4d 59 20	
w43 = w42 ⊕ w39 = 86 26 18 76	

used to generate the auxiliary word used in key expansion. We begin, of course, with the key itself serving as the round key for round 0.

Next, Table 5.4 shows the progression of State through the AES encryption process.

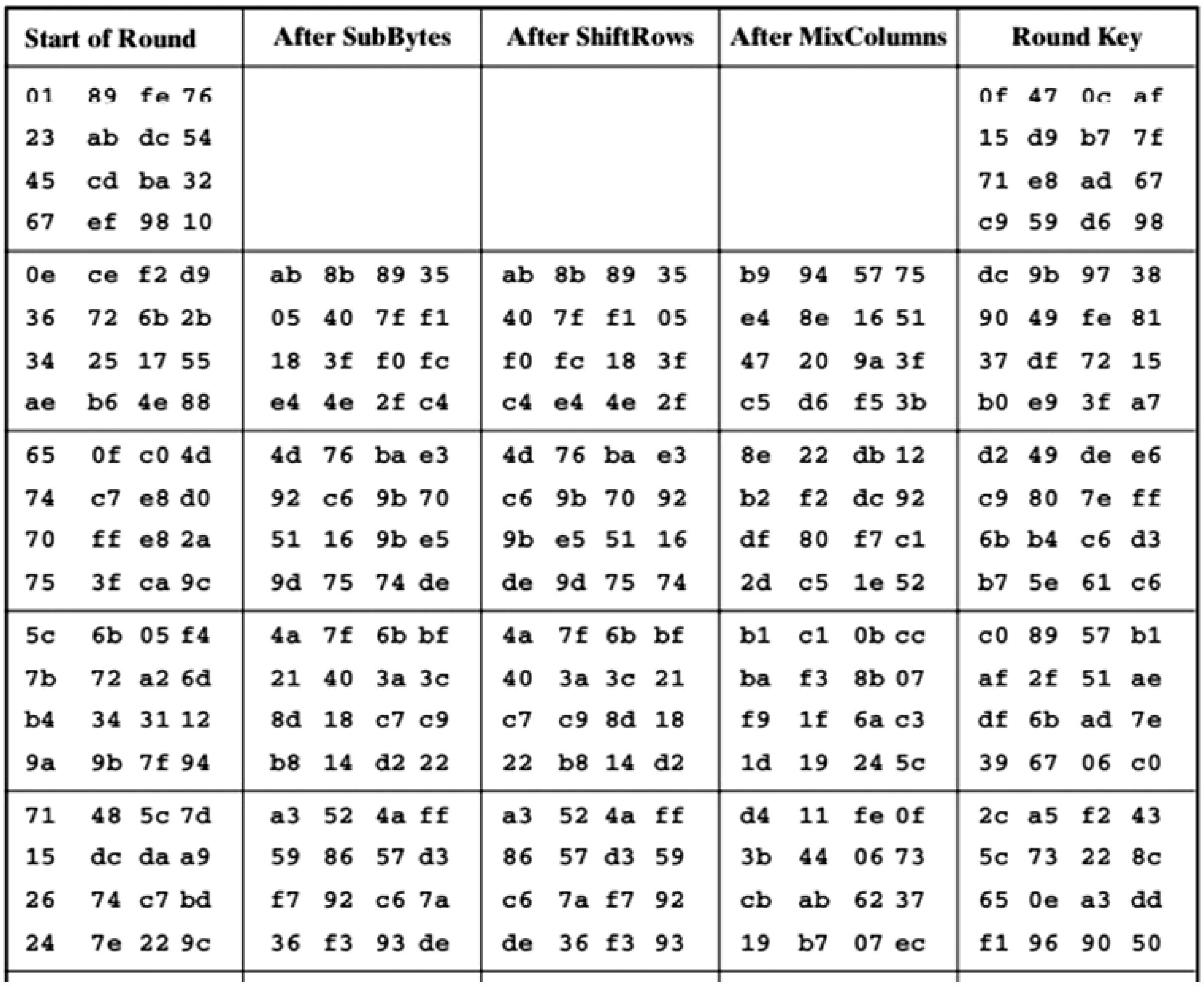
The first column showsthe value of State at the start of a round. For the first row, State is just the matrix arrangement of theplaintext.

The second, third, and fourth columns show the value of State for that round after the SubBytes,ShiftRows, and MixColumns transformations, respectively. The fifth column shows the round key. You canverify that these round keys equate with those shown in Table 5.3. The first column shows the value of Stateresulting from the bitwise XOR of State after the preceding MixColumns with the round key for the precedinground.

Avalanche Effect

If a small change in the key or plaintext were to produce a corresponding small change in the ciphertext, this might be used to effectively reduce the size of the

Table 5.4 AES Example



Start of Round	After SubBytes	After ShiftRows	After MixColumns	Round Key
01 89 fe 76 23 ab dc 54 45 cd ba 32 67 ef 98 10				0f 47 0c af 15 d9 b7 7f 71 e8 ad 67 c9 59 d6 98
0e ce f2 d9 36 72 6b 2b 34 25 17 55 ae b6 4e 88	ab 8b 89 35 05 40 7f f1 18 3f f0 fc e4 4e 2f c4	ab 8b 89 35 40 7f f1 05 f0 fc 18 3f c4 e4 4e 2f	b9 94 57 75 e4 8e 16 51 47 20 9a 3f c5 d6 f5 3b	dc 9b 97 38 90 49 fe 81 37 df 72 15 b0 e9 3f a7
65 0f c0 4d 74 c7 e8 d0 70 ff e8 2a 75 3f ca 9c	4d 76 ba e3 92 c6 9b 70 51 16 9b e5 9d 75 74 de	4d 76 ba e3 c6 9b 70 92 9b e5 51 16 de 9d 75 74	8e 22 db 12 b2 f2 dc 92 df 80 f7 c1 2d c5 1e 52	d2 49 de e6 c9 80 7e ff 6b b4 c6 d3 b7 5e 61 c6
5c 6b 05 f4 7b 72 a2 6d b4 34 31 12 9a 9b 7f 94	4a 7f 6b bf 21 40 3a 3c 8d 18 c7 c9 b8 14 d2 22	4a 7f 6b bf 40 3a 3c 21 c7 c9 8d 18 22 b8 14 d2	b1 c1 0b cc ba f3 8b 07 f9 1f 6a c3 1d 19 24 5c	c0 89 57 b1 af 2f 51 ae df 6b ad 7e 39 67 06 c0
71 48 5c 7d 15 dc da a9 26 74 c7 bd 24 7e 22 9c	a3 52 4a ff 59 86 57 d3 f7 92 c6 7a 36 f3 93 de	a3 52 4a ff 86 57 d3 59 c6 7a f7 92 de 36 f3 93	d4 11 fe 0f 3b 44 06 73 cb ab 62 37 19 b7 07 ec	2c a5 f2 43 5c 73 22 8c 65 0e a3 dd f1 96 90 50

plaintext (or key) space to be searched. What is desired is the avalanche effect, in which a small change in plaintext or key produces a large change in the ciphertext.

Using the example from Table 5.4, Table 5.5 shows the result when the eighth bit of the plaintext is changed. The second column of the table shows the value of the State matrix at the end of each round for the twoplaintexts. Note that after just one round, 20 bits of the State vector differ. After two rounds, close to half thebits differ. This magnitude of difference propagates through the remaining rounds. A bit difference inapproximately half the positions in the most desirable outcome.

f8 b4 0c 4c 67 37 24 ff ae a5 c1 ea e8 21 97 bc	41 8d fe 29 85 9a 36 16 e4 06 78 87 9b fd 88 65	41 8d fe 29 9a 36 16 85 78 87 e4 06 65 9b fd 88	2a 47 c4 48 83 e8 18 ba 84 18 27 23 eb 10 0a f3	58 fd 0f 4c 9d ee cc 40 36 38 9b 46 eb 7d ed bd
72 ba cb 04 1e 06 d4 fa b2 20 bc 65 00 6d e7 4e	40 f4 1f f2 72 6f 48 2d 37 b7 65 4d 63 3c 94 2f	40 f4 1f f2 6f 48 2d 72 65 4d 37 b7 2f 63 3c 94	7b 05 42 4a 1e d0 20 40 94 83 18 52 94 c4 43 fb	71 8c 83 cf c7 29 e5 a5 4c 74 ef a9 c2 bf 52 ef
0a 89 c1 85 d9 f9 c5 e5 d8 f7 f7 fb 56 7b 11 14	67 a7 78 97 35 99 a6 d9 61 68 68 0f b1 21 82 fa	67 a7 78 97 99 a6 d9 35 68 0f 61 68 fa b1 21 82	ec 1a c0 80 0c 50 53 c7 3b d7 00 ef b7 22 72 e0	37 bb 38 f7 14 3d d8 7d 93 e7 08 a1 48 f7 a5 4a
db a1 f8 77 18 6d 8b ba a8 30 08 4e ff d5 d7 aa	b9 32 41 f5 ad 3c 3d f4 c2 04 30 2f 16 03 0e ac	b9 32 41 f5 3c 3d f4 ad 30 2f c2 04 ac 16 03 0e	b1 1a 44 17 3d 2f ec b6 0a 6b 2f 42 9f 68 f3 b1	48 f3 cb 3c 26 1b c3 be 45 a2 aa 0b 20 d7 72 38

f9 e9 8f 2b 1b 34 2f 08 4f c9 85 49 bf bf 81 89	99 1e 73 f1 af 18 15 30 84 dd 97 3b 08 08 0c a7	99 1e 73 f1 18 15 30 af 97 3b 84 dd a7 08 08 0c	31 30 3a c2 ac 71 8c c4 46 65 48 eb 6a 1c 31 62	fd 0e c5 f9 0d 16 d5 6b 42 e0 4a 41 cb 1c 6e 56
cc 3e ff 3b a1 67 59 af 04 85 02 aa a1 00 5f 34	4b b2 16 e2 32 85 cb 79 f2 97 77 ac 32 63 cf 18	4b b2 16 e2 85 cb 79 32 77 ac f2 97 18 32 63 cf	4b 86 8a 36 b1 cb 27 5a fb f2 f2 af cc 5a 5b cf	b4 8e f3 52 ba 98 13 4e 7f 4d 59 20 86 26 18 76
ff 08 69 64 0b 53 34 14 84 bf ab 8f 4a 7c 43 b9				

Clearly, if almost all the bits are changed, this would be logically equivalent to almost none of the bits being changed. Put another way, if we selecttwo plaintexts at random, we would expect the two plaintexts to differ in about half of the bit positions andthe two ciphertexts to also differ in about half the positions.

Table 5.6 shows the change in State matrix values when the same plaintext is used and the two keys differ in the eighth bit. That is, for the second case, the key is 0e1571c947d9e8590cb7add6af7f6798.

Again, one round produces a significant change, and the magnitude of change after all subsequent rounds is roughly half the bits. Thus, based on this example, AES exhibits a very strong avalanche effect.

Table : Avalanche Effect in AES: Change in Plaintext

Round		**Number of Bits that Differ**
	0123456789abcdeddedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a672349b05a2317ff46dd1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18aeffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

Table : Avalanche Effect in AES: Change in Key

Round		**Number of Bits that Differ**
	0123456789abcdeffedcba9876543210 0123456789abcdeffedcba9876543210	0
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c5a9ad090ec7ff3fc1e8e8ca4cd02a9c	22
2	5c7bb49a6b72349b05a2317ff46d1294 90905fa9563356d15f3760f3b8259985	58
3	7115262448dc747e5cdac7227da9bd9c 18aeb7aa794b3b66629448d575c7cebf	67
4	f867aee8b437a5210c24c1974cffeabc f81015f993c978a876ae017cb49e7eec	63
5	721eb200ba06206dcbd4bce704fa654e 5955c91b4e769f3cb4a94768e98d5267	81
6	0ad9d85689f9f77bc1c5f71185e5fb14 dc60a24d137662181e45b8d3726b2920	70
7	db18a8ffa16d30d5f88b08d777ba4eaa fe8343b8f88bef66cab7e977d005a03c	74
8	f91b4fbfe934c9bf8f2f85812b084989 da7dad581d1725c5b72fa0f9d9d1366a	67
9	cca104a13e678500ff59025f3bafaa34 0ccb4c66bbfd912f4b511d72996345e0	59
10	ff0b844a0853bfc6934ab4364148fb9 fc8923ee501a7d207ab670686839996b	3

2.1.4 AES Implementation

Q6. Explain about the applications of AES.

Ans :

Equivalent Inverse Cipher

That is, the sequence of transformations for decryption differs from that for encryption, although the form of the keysschedules for encryption and decryption is the same. This has the disadvantage that two separate softwareor firmware modules are needed for applications that require both encryption and decryption. There is,however, an equivalent version of the decryption algorithm that has the same structure as the encryptionalgorithm.

Two separate changes are needed to bring the decryption structure in line with the encryption structure. The standard decryption round has the structure InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns. Thus, the first two stages of the decryption round need to be interchanged, and the second two stages of the decryption round need to be interchanged.

Interchanging Invshiftrows and Invsubbytes : InvShiftRows affects the sequence of bytes in State butdoes not alter byte contents and does not depend on byte contents to perform its transformation. InvSubBytes affects the contents of bytes in State but does not alter byte sequence and does not depend on byte sequence to perform its transformation. Thus, these two operations commute and can be interchanged.For a given State Si,

InvShiftRows [InvSubBytes (Si)] = InvSubBytes [InvShiftRows (Si)]

Interchanging Addroundkey and Invmixcolumns : The transformations Add- RoundKey and InvMixColumns do not alter the sequence of bytes in State. If we view the key as a sequence of words, thenboth AddRoundKey and InvMixColumns operate on State one column at a time. These two operations arelinear with respect to the column input. That is, for a given State Si and a given round key wj,

InvMixColumns (Si Í$ wj) = [InvMixColumns (Si)] Í$ [InvMixColumns (wj)]

To see this, suppose that the first column of State Si is the sequence is the sequence (y0, y1, y2, y3) and the first column of the round key wj is (k0, k1, k2, k3). Then we need to show

$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 & \oplus & k_0 \\ y_1 & \oplus & k_1 \\ y_2 & \oplus & k_2 \\ y_3 & \oplus & k_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} + \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} k_0 \\ k_1 \\ k_2 \\ k_3 \end{bmatrix}$$

Let us demonstrate that for the first column entry. We need to show

$$\left[\{0E\} \bullet (y_0 \oplus k_0)\right] \oplus \left[\{0B\} \bullet (y_1 \oplus k_1)\right] \oplus \left[\{0D\} \bullet (y_2 \oplus k_1)\right] \oplus \left[\{09\} \bullet (y_3 \oplus k_3)\right]$$

$$= \left[\{0E\} \bullet y_0\right] \oplus \left[\{0B\} \bullet y_1\right] \oplus \left[\{0D\} \bullet y_2\right] \oplus \left[\{09\} \bullet y_3\right] \oplus$$

$$= \left[\{0E\} \bullet y_0\right] \oplus \left[\{0B\} \bullet y_1\right] \oplus \left[\{0D\} \bullet y_2\right] \oplus \left[\{09\} \bullet y_3\right]$$

This equation is valid by inspection.Thus, we can interchange AddRoundKey and InvMixColumns, provided that we first apply InvMixColumns to the round key. Note that we do not need to apply InvMixColumns to theround key for the input to the first AddRoundKey transformation nor to the lastAddRoundKey transformation.This is because these two AddRoundKey transformations are notinterchanged with InvMixColumns to produce the equivalent decryption algorithm.

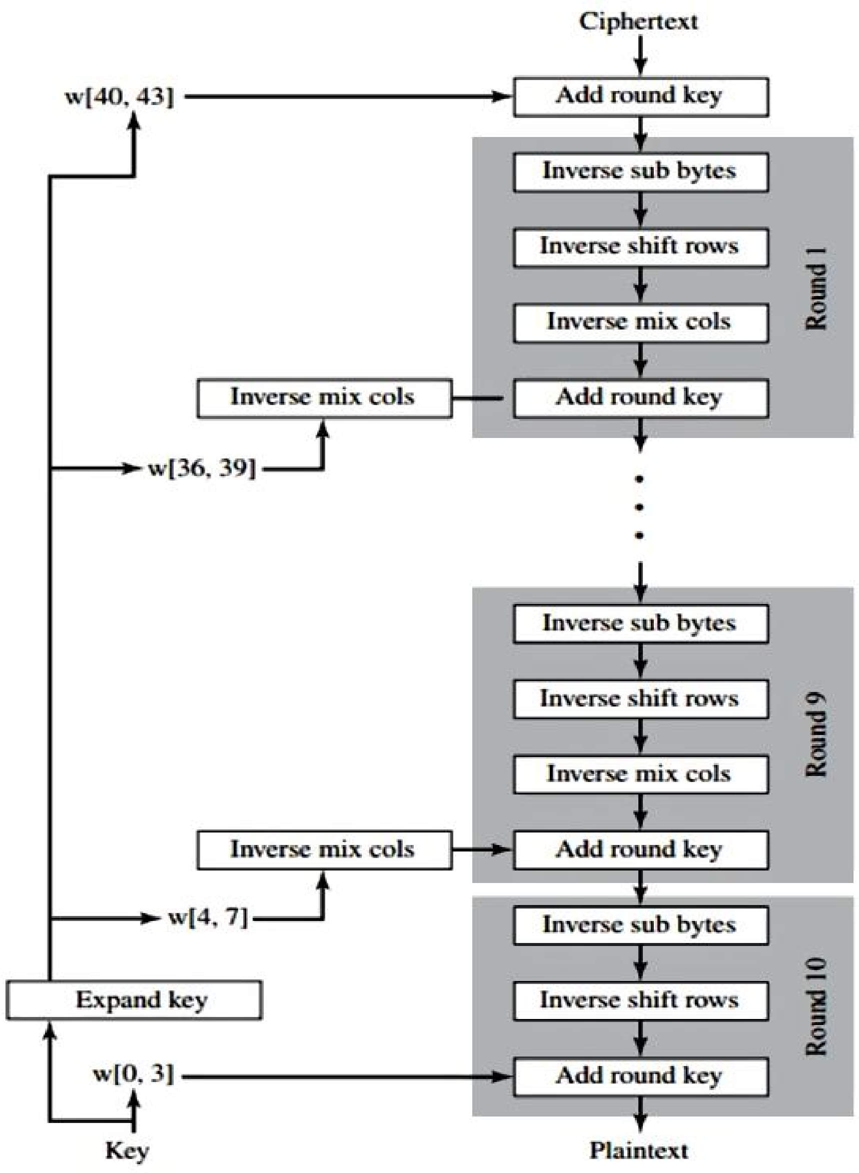


Fig. : illustrates the equivalent decryption algorithm

8-Bit Processor

AES can be implemented very efficiently on an 8-bit processor. AddRoundKey is a bytewise XOR operation. ShiftRows is a simple byte-shifting operation. SubBytes operates at the byte leveland only requires a table of 256 bytes. The transformation MixColumns requires matrix multiplication in thefield GF(28), which means that all operations are carried out on bytes.

The transformation MixColumns requires matrix multiplication in the field GF(28), which means that alloperations are carried out on bytes. MixColumns only requires multiplication by {02} and {03}, which, as wehave seen, involved simple shifts, conditional XORs, and XORs. This can be implemented in a moreefficient way that eliminates the shifts and conditional XORs. Equation set (5.4) shows the equations for the MixColumns transformation on a single column. Using the identity {03} x = ({02} x) ⊗ x, we can rewrite Equation set [5.4] as follows :

$$\begin{aligned}
Tmp &= s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\
s'_{0,j} &= s_{0,j} \oplus Tmp \oplus [2 \bullet (s_{0,j} \oplus s_{1,j})] \\
s'_{1,j} &= s_{1,j} \oplus Tmp \oplus [2 \bullet (s_{1,j} \oplus s_{2,j})] \\
s'_{2,j} &= s_{2,j} \oplus Tmp \oplus [2 \bullet (s_{2,j} \oplus s_{3,j})] \\
s'_{3,j} &= s_{3,j} \oplus Tmp \oplus [2 \bullet (s_{3,j} \oplus s_{0,j})]
\end{aligned} \qquad \textbf{(5.9)}$$

The multiplication by {02} involves a shift and a conditional XOR. Such an implementation may be vulnerable to a timing attack of the sort described in Section 3.4. To counter this attack and to increase processing efficiency at the cost of some storage, the multiplication can be replaced by a table lookup. Define the 256-byte table X2, such that X2[i] = {02} • i. Then Equation set (5,9) can be rewritten as

$$\begin{aligned}
Tmp &= s_{0,j} \oplus s_{1,j} \oplus s_{2,j} \oplus s_{3,j} \\
s'_{0,j} &= s_{0,j} \oplus Tmp \oplus X2[s_{0,j} \oplus s_{1,j}] \\
s'_{1,c} &= s_{1,j} \oplus Tmp \oplus X2[s_{1,j} \oplus s_{2,j}] \\
s'_{2,c} &= s_{2,j} \oplus Tmp \oplus X2[s_{2,j} \oplus s_{3,j}] \\
s'_{3,j} &= s_{3,j} \oplus Tmp \oplus X2[s_{3,j} \oplus s_{0,j}]
\end{aligned}$$

32-BIT PROCESSOR. Fora 32-bit processor, a more efficient implementation can be achieved if operations are defined on 32-bit words.To show this, we first define the four transformations of a round in algebraic form. Suppose we begin with a State matrix consisting of elements ai,j and a round-key matrix consisting of elements ki, j. Then the transformations can be expressed as follows.

Sub Bytes	$b_{i,j} = S[a_{i,j}]$
Shift Rows	$\begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix} = \begin{bmatrix} b_{0,j} \\ b_{1,j-1} \\ b_{2,j-2} \\ b_{3,j-3} \end{bmatrix}$
Mix Columns	$\begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} c_{0,j} \\ c_{1,j} \\ c_{2,j} \\ c_{3,j} \end{bmatrix}$
AddRoundKey	$\begin{bmatrix} e_{0,j} \\ e_{1,j} \\ e_{2,j} \\ e_{3,j} \end{bmatrix} = \begin{bmatrix} d_{0,j} \\ d_{1,j} \\ d_{2,j} \\ d_{3,j} \end{bmatrix} \oplus \begin{bmatrix} k_{0,j} \\ k_{1,j} \\ k_{2,j} \\ k_{3,j} \end{bmatrix}$

In the ShiftRows equation, the column indices are taken mod 4. We can combine all of these expressions into a single equation :

$$\begin{bmatrix} e_{0.j} \\ e_{1.j} \\ e_{2.j} \\ e_{3.j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S[a_{0.j}] \\ S[a_{1.j-1}] \\ S[a_{2.j-2}] \\ S[a_{3.j-3}] \end{bmatrix} \oplus \begin{bmatrix} k_{0.j} \\ k_{1.j} \\ k_{2.j} \\ k_{3.j} \end{bmatrix}$$

$$= \left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \bullet S[a_{0.j}] \right) \oplus \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \bullet S[a_{1.j-1}] \right) \oplus \left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \bullet S[a_{2.j-2}] \right)$$

$$\oplus \left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \bullet S[a_{3.j-3}] \right) \oplus \begin{bmatrix} k_{0.j} \\ k_{1.j} \\ k_{2.j} \\ k_{3.j} \end{bmatrix}$$

In the second equation, we are expression the matrix multiplication as a linear combination of vectors. Wedefine four 256-word (1024-byte) tables as follows

$T_0[x]\left(\begin{bmatrix} 02 \\ 01 \\ 01 \\ 03 \end{bmatrix} \bullet S[x] \right)$	$T_1[x]\left(\begin{bmatrix} 03 \\ 02 \\ 01 \\ 01 \end{bmatrix} \bullet S[x] \right)$	$T_2[x]\left(\begin{bmatrix} 01 \\ 03 \\ 02 \\ 01 \end{bmatrix} \bullet S[x] \right)$	$T_3[x]\left(\begin{bmatrix} 01 \\ 01 \\ 03 \\ 02 \end{bmatrix} \bullet S[x] \right)$

Thus, each table takes as input a byte value and produces a column vector (a 32-bit word) that is a function of the S-box entry for that byte value. These tables can be calculated in advance.

We can define a round function operating on a column in the following fashion.

$$\begin{bmatrix} S'_{0.j} \\ S'_{1.j} \\ S'_{2.j} \\ S'_{3.j} \end{bmatrix} = T_0[S_{0.j}] \oplus T_1[S_{1.j-1}] \oplus T_2[S_{2.j-2}] \oplus T_3[S_{3.j-3}] \oplus \begin{bmatrix} k_{0.j} \\ k_{1.j} \\ k_{2.j} \\ k_{3.j} \end{bmatrix}$$

As a result, an implementation based on the preceding equation requires only four table lookups and fourXORs per column per round, plus 4 Kbytes to store the table. The developers of Rijndeal believe that this compact, efficient implementation was probably one of the most important factors in the selction of Rijndaelfor AES.

2.2 PSEUDO RANDOM NUMBER GENERATION AND STREAM CIPHERS

2.2.1 Principles of PSEUDO Random Number Generation

Q7. Explain the principles of pseudo random number generation.

Ans :

1. The Use of Random Numbers

A number of network security algorithms and protocols based on cryptography make use of random binary numbers. For example,

- Theuse of random numbers for the nonces frustrates an opponent's efforts to deter- mine or guess the nonce.
- Session key generation. We will see a number of protocols in this book where a secret key forsymmetric encryption is generated for use for a short period of time. This key is generally called a session key.
- Generation of keys for the RSA public-key encryption algorithm
- Generation of a bit stream for symmetric stream .

These applications give rise to two distinct and not necessarily compatible requirements for a sequence ofrandom numbers: randomness and unpredictability.

2. Randomness

Traditionally, the concern in the generation of a sequence of allegedly random numbers has been that the sequence of numbers be random in some well- defined statistical sense. The following twocriteria are used to validate that a sequence of numbers is random:

- Uniform distribution: The distribution of bits in the sequence should be uniform; that is, thefrequency of occurrence of ones and zeros should be approximately equal.
- Independence: No one subsequence in the sequence can be inferred from the others.

3. Unpredictability

In applications such as reciprocal authentication, session key generation, and stream ciphers, the requirement is not just that the sequence of numbers be statistically random but that the successive members of the sequence are unpredictable. With "true" random sequences, each number isstatistically independent of other numbers in the sequence and therefore unpredictable.

4. TRNGs, PRNGs, and PRFs

Cryptographic applications typically make use of algorithmic techniques for random number generation.These algorithms are deterministic and therefore produce sequences of numbers that are not statisticallyrandom. However, if the algorithm is good, the resulting sequences will pass many reasonable tests ofrandomness. Such numbers are referred to as pseudorandom numbers.

Figure contrasts a true random number generator (TRNG) with two forms of pseudorandom numbergenerators.

A TRNG takes as input a source that is effectively random; the source is often referred to as anentropy source. The source, orcombination of sources, serve as input to an algorithm that produces random binary output. The TRNG maysimply involve conversion of an analog source to a binary output. The TRNG may involve additionalprocessing to overcome any bias in the source;

In contrast, a PRNG takes as input a fixed value, called the seed, and produces a sequence of output bits using a deterministic algorithm. Typically, as shown, there is some feedback path by which some of the results of the algorithm are fed back as input as additional output bits are produced. The important thing to note is that the output bit stream isdetermined solely by the input value or values, so that an adver- sary who knows the algorithm and the seedcan reproduce the entire bit stream.

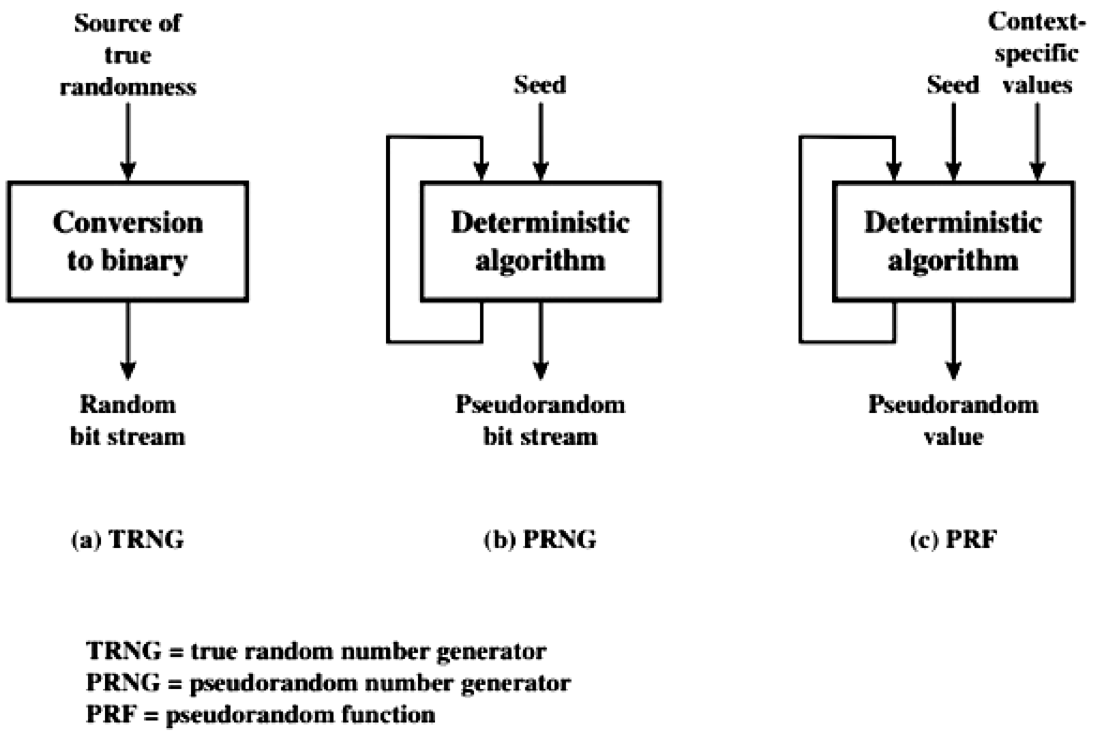


Fig. : Random and Pseudorandom Number Generators

- Pseudorandom number generator: An algorithm that is used to produce an open-ended sequenceof bits is referred to as a PRNG. A common application for an open-ended sequence of bits is as input to asymmetric stream cipher,
- Pseudorandom function (PRF): A PRF is used to produced a pseudorandom string of bits of some fixed length. Examples are symmetric encryption keys and nonces. Typically, the PRF takes as input a seedplus some context specific values, such as a user ID or an application ID.

5. Seed Requirements

For cryptographic applications, the seed that serves as input to the PRNG must be secure. Because the PRNG is a deterministic algorithm, if the adversary can deduce the seed, then the output can also be determined. Therefore, the seed must be unpredictable. In fact, the seed itself must be arandom or pseudorandom number.

Typically, the seed is generated by a TRNG, as shown in Figure 7.2. if a TRNG is available, why it is necessary to use a PRNG. If theapplication is a stream cipher, then a TRNG is not practical. The sender would need to generate akeystream of bits as long as the plaintext and then transmit the keystream and the ciphertext securely to the receiver. If a PRNG is used, the sender need only find a way to deliver the stream cipher key, which istypically 54 or 128 bits, to the receiver in a secure fashion.

Even in the case of a PRF application, in which only a limited number of bits is generated, it is generally desirable to use a TRNG to provide the seed to the PRF and use the PRF output rather then use the TRNG directly.

Finally, the mechanism used to generate true random numbers may not be able to generate bits at a rate sufficient to keep up with the application requiring the random bits.

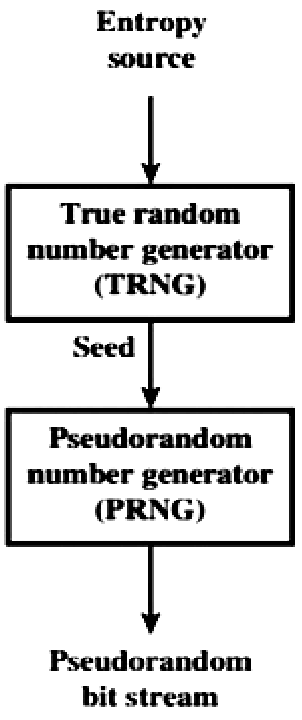

Entropy source
True random number generator (TRNG)
Seed
Pseudorandom number generator (PRNG)
Pseudorandom bit stream

Fig. : Generation of Seed Input to PRNG

Algorithm Design

Cryptographic PRNGs have been the subject of much research over the years, and a wide variety of algorithms have been developed. These fall roughly into two categories.

- Purpose-built algorithms: These are algorithms designed specifically and solely for the purpose ofgenerating pseudorandom bit streams. Some of these algorithms are used for a variety of PRNG applications.
- Algorithms based on existing cryptographic algorithms: Cryptographic algorithms have the effect of randomizing input. Indeed, this is a requirement of such algorithms. For example, if a symmetricblock cipher produced ciphertext that had certain regular patterns in it, it would aid in the process ofcryptanalysis. Thus, cryptographic algorithms can serve as the core of PRNGs. Three broad categories ofcryptographic algorithms are commonly used to create PRNGs:

— Symmetric block ciphers

— Asymmetric ciphers

— Hash functions

Any of these approaches can yield a cryptographically strong PRNG. A purpose-built algorithm may be provided by an operating system for general use. For applications that already use certain cryptographicalgorithms for encryption or authentication, it makes sense to reuse the same code for the PRNG. Thus, all of these approaches are in common use.

2.2.2 Pseudorandom Number Generators

Q8. Explain about various pseudo random generators.

Ans :

Linear Congruential Generators

A widely used technique for pseudorandom number, which is known as the linear congruential method. The algorithm is parameterized with fournumbrs, as follows :

m	The modulus	M > 0
a	The multiplier	$0 < a < m$
c	The increment	$0 \le c < m$
X_0	The starting value, or seed	$0 \le X_0 < m$

The sequence of random numbers $\{Xn\}$ is obtained via the following iterative equation:

$$Xn + 1 = (aXn + c)\text{mod } m$$

If m, a, c, and X0 are integers, then this technique will produce a sequence of integers with each integer in the range 0 … Xn 6 m.

The selection of values for a, c, and m is critical in developing a good random number generator. For example, consider a = c = 1. The sequence produced is obviously not satisfactory. Now consider the values a = 7, c = 0, m = 32, and X0 = 1. This generates the sequence 57, 17, 23, 1, 7, etc.6, which is also clearly unsatisfactory. Of the 32 possible values, only four are used; thus, the sequence is said to have a period of 4. If, instead, we change the value of a to 5, then the sequence is 55, 25, 29, 17, 21, 9, 13, 1, 5, etc.6, which increases the period to 8.

We would like m to be very large, so that there is the potential for producing a long series of distinct random numbers. A common criterion is that m be nearly equal to the maximum representable nonnegative integer for a given computer. Thus, a value of m near to or equal to 231 is typically chosen.

Three tests to be used in evaluating a random number generator:

T1: The function should be a full-period generating function. That is, the function should generate all the numbers between 0 and m before repeating.

T2: The generated sequence should appear random.

T3: The function should implement efficiently with 32-bit arithmetic.

With appropriate values of a, c, and m, these three tests can be passed. With respect to T1, it can be shownthat if m is prime and c = 0, then for certain values of a the period of the generating function is m - 1, withonly the value 0 missing. For 32- bit arithmetic, a convenient prime value of m is 231 - 1. Thus, the generatingfunction becomes

$X_n + 1 = (aX_n) \bmod (231 - 1)$

The strength of the linear congruential algorithm is that if the multiplier and modulus are properly chosen, the resulting sequence of numbers will be statistically indistinguishable from a sequence drawn at random from the set 1, 2, Á , m - 1. But there is nothing random at all about the algorithm, apart from the choice of the initial value X0. Once that value is chosen, the remain- ing numbers in the sequence follow deterministically. This has implications for cryptanalysis.

If an opponent knows that the linear congruential algorithm is being used and if the parameters are known(e.g., a = 75, c = 0, m = 231 - 1), then once a single number is discovered, all subsequent numbers are known. Even if the opponent knows only that a linear congruential algorithm is being used, knowledge of a small part of the sequence is sufficient to determine the parameters of the algorithm. Suppose that the opponent is able to determine values for X0, X1, X2, and X3. Then

$X1 = (aX0 + c) \bmod m$

$X2 = (aX1 + c) \bmod m$

$X3 = (aX2 + c) \bmod m$

These equations can be solved for a, c, and m.

Blum Blum Shub Generator

A popular approach to generating secure pseudorandom numbers is known as the Blum, Blum, Shub (BBS) generator. It has perhaps the strongest public proof of its cryptographicstrength of any purpose-built algorithm. The procedure is as follows. First, choose two large prime numbers, pand q, that both have a remainder of 3 when divided by 4. That is,

p K q K 3(mod 4)

simply means that (p mod 4) = (q mod 4) = 3.

For example, the prime numbers 7 and 11 satisfy 7 K 11 K 3(mod 4).

Let n = p * q. Next, choose a random number s, such that s is relatively prime to n; this is equivalent to saying that neither p nor q is a factor of s.

Then the BBS generator produces a sequence of bits Bi according to the following algorithm:

```
X0 = s2 mod n
for i =  1 to q
Xi = (Xi - 1)2 mod n
Bi = Xi  mod 2
```

Thus, the least significant bit is taken at each iteratin. Table 7.1, shows an example of BBS operation. Here, n = 192649 = 383 * 503, and the seed s = 101355

The BBS is referred to as a cryptographically secure pseudorandom bit generator (CSPRBG). A pseudorandom bit generator is said to pass the next-bit test if there is not a polynomial-time algorithm2that, on.

Input of the first k bits of an output sequence, can predict the [k + 1)st bit with probability significcant grerater than 1/2. In other words, given the first k bits of the sequence, there is not a practical algorithm that can even allow you to state that the next bit will be 1 [or 0] with probability greater than 1/2 Table 7.1 Example operation of BBS Generator

I	X_i	B_i	I	X_i	B_i
0	20749		11	137922	0
1	143135	1	12	123175	1
2	177671	1	13	8630	0
3	97048	0	14	114386	0
4	89992	0	15	14863	1
5	174051	1	16	133015	1
6	80649	1	17	106065	1
7	45663	1	18	45870	0
8	69442	0	19	137171	1
9	186894	0	20	48060	0
10	177046	0			

2.2.3 Pseudorandom Number Generation using a Block Cipher

Q9. Explain, how to generate a Pseudo Random number using a Block Cipher ?

Ans :

A popular approach to PRNG construction is to use a symmetric block cipher as the heart of the PRNGmechanism. For any block of plaintext, a symmetric block cipher produces an output block that is apparentlyrandom. That is, there are no patterns or regularities in the ciphertext that provide information that can be used to deduce the plaintext. Thus, a symmetric block cipher is a good candidate for building a pseudorandom number generator.

PRNG Using Block Cipher Modes of Operation

Two approaches that use a block cipher to build a PNRG: the CTRmode and the OFB mode.

Figure 7.3 illustrates the two methods. In each case, the seed consists of two parts: the encryption key value and a value V that will be updated after each block of pseudorandom numbers is generated. Thus, for AES-128, the seed consists of a 128-bit key and a 128-bit V value. In the CTR case, the value of V is incremented by 1 after each encryption. In the case of OFB, the value of V is updated to equal the value ofthe preceding PRNG block. In both cases, pseudorandom bits are produced one block at a time (e.g., forAES, PRNG bits are generated 128 bits at a time).

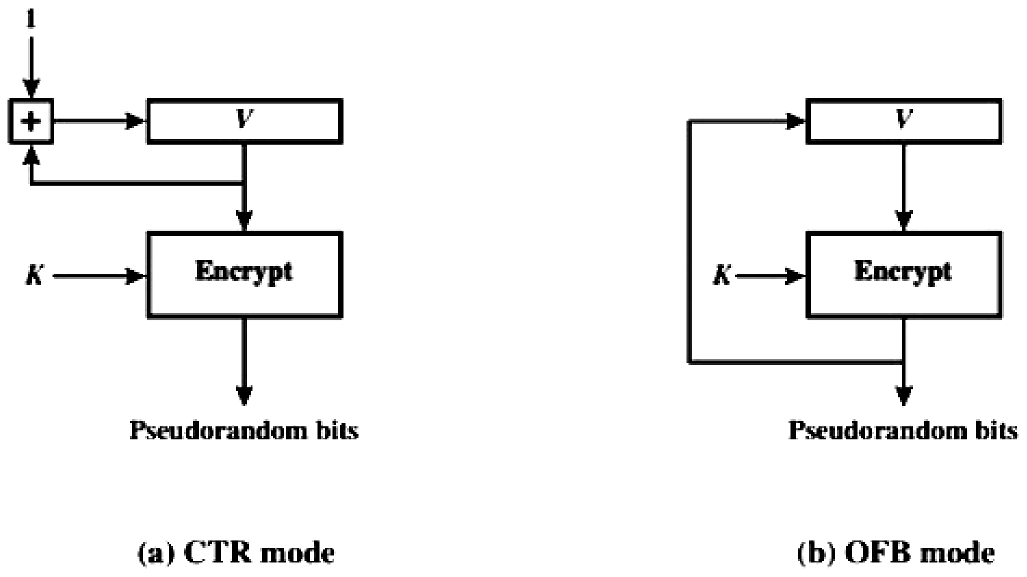


Fig. : PRNG Mechanisms Based on Block Ciphers

The CTR algorithm for PRNG can be summarized as follows.

```
While (len (temp) < requested_number_of_bits) do
V = (V + 1) mod 2128.
output_block = E(Key, V) temp = temp || ouput_block
```

The OFB algorithm can be summarized as follows.

```
While (len (temp) < requested_number_of_bits) do
V = E(Key, V)
temp = temp || V
```

To get some idea of the performance of these two PRNGs, consider the fol- lowing short experiment. Arandom bit sequence of 256 bits was obtained from random.org, which uses three radios tuned betweenstations to pick up atmospheric noise. These 256 bits form the seed, allocated as

Key : cfb0ef3108d49cc4562d5810b0a9af60

V : 4c89af496176b728ed1e2ea8ba27f5a4

The total number of one bits in the 256-bit seed is 124, or a fraction of 0.48, which is reassuringly close to the ideal of 0.5.

For the OFB PRNG, Table 7.2 shows the first eight output blocks (1024 bits) with two rough measures ofsecurity. The second column shows the fraction of one bits in each 128-bit block. This corresponds to one ofthe NIST tests. The results indi- cate that the output is split roughly equally between zero and one bits. The third column shows the fraction of bits that match between adjacent blocks. If this number differssubstantially from 0.5, that suggests a correlation between blocks, which could be a security weakness. Theresults suggest no correlation.

Table : Example Results for PRNG Using OFB

Output Block	Fraction of One Bits	Fraction of Bits that Match with Proceeding Block
1786f4c7ff6e291dbdfdd90ec3453176	0.57	-
5e17b22b14677a4d66890f87565eae64	0.51	0.52
fd18284ac82251dfb3aa62c326cd46cc	0.47	0.54
c8e545198a758ef5dd86b41946389bd5	0.50	0.44
fe7bae0e23019542962e2c52d215a2e3	0.47	0.48
14fdf5ec99469598ae0379472803accd	0.49	0.52
6aeca972e5a3ef17bd1a1b775fc8b929	0.57	0.48
f7e97badf359d128f00d9b4ae323db64	0.55	0.45

S

1786f4c7ff6e291dbdfdd90ec3453176	0.57	-
60809669a3e092a01b463472fdcae420	0.41	0.41
d4e6e170b46b0573eedf88ee39bff33d	0.59	0.45
5f8fcfc5deca18ea246785d7fadc76f8	0.59	0.52
90e63ed27bb07868c753545bdd57ee28	0.53	0.52
0125856fdf4a17f747c7833695c52235	0.50	0.47
F4be2d179b0f2548fd748c8fc7c81990	0.51	0.48
1151fc48f90eabac658a3911515c3c66	0.47	0.45

Table : shows the results using the same key and *V* values for CTR mode.

2.2.4 Stream Ciphers

Q9. What are stream ciphers ? Explain about it.

Ans :

A typical stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed tooperate on one bit at a time or on units larger than a byte at a time. Figure 7.5 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. The output of the generator, called a keystream, is combined one byte at a time with the plaintext stream using the bit-wise 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
```

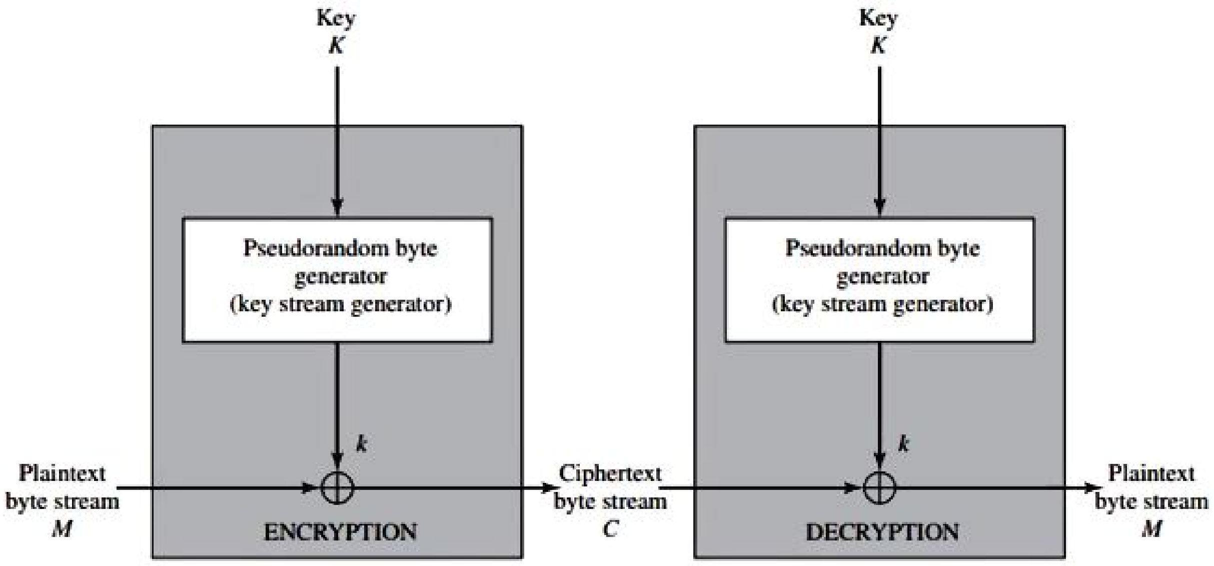

Key
K
Pseudorandom byte generator (key stream generator)
k
Plaintext byte stream M
ENCRYPTION
Ciphertext byte stream C
Key
K
Pseudorandom byte generator (key stream generator)
k
DECRYPTION
Plaintext byte stream M

Figure : Stream Cipher Diagram

Descryption requires the use of the same pseudorandom sequence.

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

The stream cipher is similar to the one-time pad . The difference is that a one-timepad uses a genuine random number stream, whereas a stream cipher uses a pseudorandom number stream.

lists the following important design considerations for a stream cipher.

1. The encryption sequence should have a large period. A pseudorandom number generator uses a functionthat produces a deterministic stream of bits that eventually repeats. The longer the period of repeat the moredifficult it will be to do cryptanalysis.

2. The keystream should approximate the properties of a true random number stream as close as possible. Forexample, there should be an approximately equal number of 1s and 0s. If the keystream is treated as a stream ofbytes, then all of the 256 possible byte values should appear approximately equally often. The more random-appearing the keystream is, the more randomized the ciphertext is, making cryptanalysis more difficult.

3. Note from Figure 7.5 that the output of the pseudorandom number generator is conditioned on the valueof the input key. To guard against brute-force attacks, the key needs to be sufficiently long. The same considerations that apply to block ciphers are valid here. Thus, with current technology, a key length of atleast 128 bits is desirable.

Cipher	Key Length	Speed (Mbps)
DES	56	9
3DES	168	3
RC2	Variable	0.9
RC4	Variable	45

With a properly designed pseudorandom number generator, a stream cipher can be as secure as a blockcipher of comparable key length. A potential advantage of a stream cipher is that stream ciphers that do notuse block ciphers as a building block are typically faster and use far less code than do block ciphers.

One advantage of a block cipher is that you can reuse keys. In contrast, if two plaintexts are encrypted with the same key using a stream cipher, then cryptanalysis is often quite simple. If the two ciphertext streams areXORed together, the result is the XOR of the original plaintexts. If the plaintexts are text strings, credit card numbers, or other byte streams with known properties, then cryptanalysis may be successful.

2.2.5 RC4

Q10. Explain about RC4 stream Cipher.

Ans :

RC4 is a stream cipher designed for RSA Security. It is a variable key size streamcipher with byte-oriented operations. The algorithm is based on the use of a random permutation. Analysisshows that the period of the cipher is overwhelmingly likely to be greater than 10100 . Eight tosixteen machine operations are required per output byte, and the cipher can be expected to run very quickly in software.

RC4 is used in the Secure Sockets Layer/ Transport Layer Security (SSL/TLS) standards that have been defined for communication between Web browsers and servers. It is also used in the WiredEquivalent Privacy (WEP) protocol and the newer WiFi Protected Access (WPA) protocol that are part ofthe IEEE wireless LAN standard.

The RC4 algorithm is remarkably simple and quite easy to explain. A variable-length key of from 1 to 256bytes (8 to 2048 bits) is used to initialize a 256-byte state vector S, with elements S[0], S[1], Á , S[255].

Initialization of S

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

//* **Initialization** */ **for** i = 0 **to** 255 **do** S[i] = i;

T[i] = K[i **mod** keylen];

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

/* **Initial Permutation of S** */ j = 0;

for i = 0 **to** 255 **do**

j = (j + S[i] + T[i]) **mod** 256;

Swap (S[i], S[j]);

Because the only operation on S is a swap, the only effect is a permutation.

S still contains all the numbers from 0 through 255.

Stream Generation

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

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

To encrypt, XOR the value k with the next byte of plain text. to decrypt, XOR the value k with the next byte of ciphertext.

Strength of RC4

A number of papers have been published analyzing methods of attacking RC4 (e.g., [KNUD98],[MIST98], [FLUH00], [MANT01]). None of these approaches is practical against RC4 with a reasonable keylength, such as 128 bits.

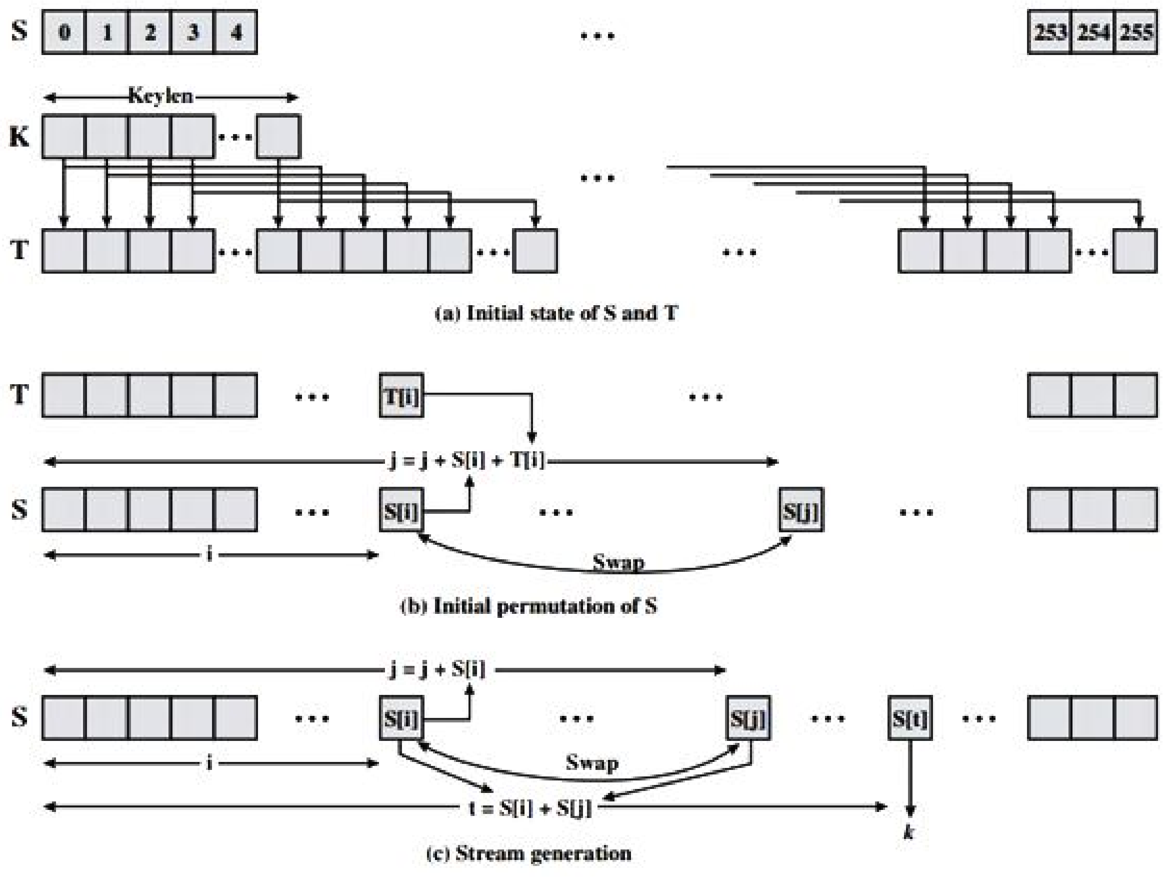


Figure : RC4

This problem points out the difficulty in designing a secure system that involves both cryptographic functions and protocols that make use of them.

2.3 Public Key Cryptography and RSA

2.3.1 Principles of Public Key Cryptosystems

Q11. Explain various principles of public key cryptosystems.

Ans :

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

Key distribution under symmetric key encryption requires either (1) that two communicants already share a key, which someone has been distributed to them or (2) the use of a key distribution center.

- Digital signatures.

1. Public key cryptosystems

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

These algorithms have the following important characteristics:

- It is computationally infeasible to determine the decryption key given only the 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.

The essential steps are the following:

- Each user generates a pair of keys to be used for encryption and decryption of messages.
- 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.
- If A wishes to send a confidential message to B, A encrypts the message using B s public key.
- When B receives the message, it decrypts using its private key. No other recipient can decrypt the message because only B knows B 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 not be distributed. As long as a system controls its private key, its incoming communication is secure.

Let the plaintext be X = [X1, X2, X3, …,Xm] where m is the number of letters in some finite alphabets. Suppose A wishes to send a message to B. B generates a pair of keys: a public key KU_b and a private key KR_b. KR_b is known only to B, whereas KU_b is publicly available and therefore accessible by A.

With the message X and encryption key KU_b as input, A forms the cipher text

Y = [Y1, Y2, Y3, ... Yn]., i.e., Y = E KU_b(X)

The receiver can decrypt it using the private key KR_b. i.e., X = D KR_b(). The encrypted message serves as a**digital signature.**

It is important to emphasize that the encryption process just described does not provide confidentiality. There is no protection of confidentiality because any observer can decrypt the message by using the sender s public key.

It is however, possible to provide both the authentication and confidentiality by a double use of the public scheme.

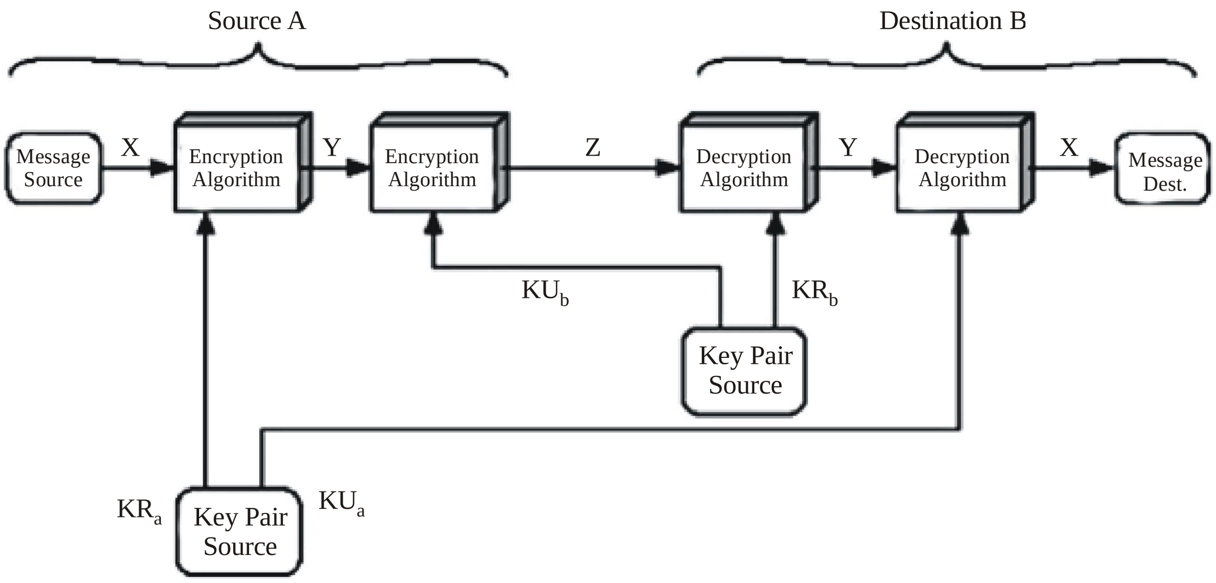


Fig. : Public Key Cryptosystem

Chipertext Z = EKU_b [EKR_a (X)]

Plaintext X = EKU_a [EKR_b (Y)]

Initially, the message is encrypted using the sender s private key. This provides the digital signature. Next, we encrypt again, using the receiver s public key. The final ciphertext can be decrypted only by the intended receiver, who alone has the matching private key. Thus confidentiality is provided.

2. Requirements for public key cryptography

- It is computationally easy for a party B to generate a pair [KU_b, KR_b].
- It is computationally easy for a sender A, knowing the public key and the message to be encrypted M, to generate the corresponding ciphertext: C=EKU_b(M).
- It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message: M = DKR_b (C) = DKR_b [EKU_b (M)]
- It is computationally infeasible for an opponent, knowing the public key KU_b, to determine the private key KR_b.
- It is computationally infeasible for an opponent, knowing the public key KU_b, and a ciphertext C, to recover the original message M.
- The encryption and decryption functions can be applied in either order: ? M = EKU_b [DKR_b (M) = DKU_b[EKR_b (M)]

Public key cryptanalysis

Public key encryption scheme is vulnerable to a brute force attack. The counter measure is to use large keys.

2.3.2 RSA Algorithm

Q12. Explain briefly RSA Algorithm.

Ans :

It was developed by Rivest, Shamir and Adleman. This algorithm 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); in practice, the block size is k-bits, where $2^k < n < 2^{k+1}$ Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C:

$C = M^e \mod n$

$= C^{d \mod} n = (M^e \mod n) \mod n$

$(M^e)^d \mod n = M^{ed} \mod n$

Both the sender and receiver 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 of KU = {e, n} and a private key of KR = {d, n}. For this algorithm to be satisfactory for public key encryption, the following requirements must be met:

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

Let us focus on the first requirement. We need to find the relationship of the form:

$$M^{ed =} M \mod n$$

A corollary to Euler s theorem fits t0068e bill: Given two prime numbers p and q and two integers, n and m, such that n=pq and 0<m<n, and arbitrary integer k, the following relationship holds

$$_m k\Phi(n) + 1 = {}_m k(p-1)(q-1) + 1 = {}_{m \mod n}$$

where Φ (n) – Euler totient function, which is the number of positive integers less than n and relatively prime to n. we can achieve the desired relationship, if $ed = k\Phi(n)+1$

This is equivalent to saying:

$ed\ a \equiv 1 \mod \Phi(n)\ d = e^{-1} \mod \Phi(n)$

That is, e and d are multiplicative inverses mod Ô(n). According to the rule of modular arithmetic, this is true only if d (and therefore e) is relatively prime to Ô(n). Equivalently,

$\gcd(\Phi(n), d) = 1.$

The steps involved in RSA algorithm for generating the key are

- Select two prime numbers, p = 17 and q = 11.
- Calculate n = p*q = 17*11 = 187
- Calculate Ô(n) = (p-1)(q-1) = 16*10 = 160.
- Select e such that e is relatively prime to Ô(n) = 160 and less than Ô(n); we choose e = 7
- Determine d such that ed a” 1 mod Ô(n) and d<160. the correct value is d = 23, because

$$23*7 = 161 = 1 \mod 160$$

1. Key Generation

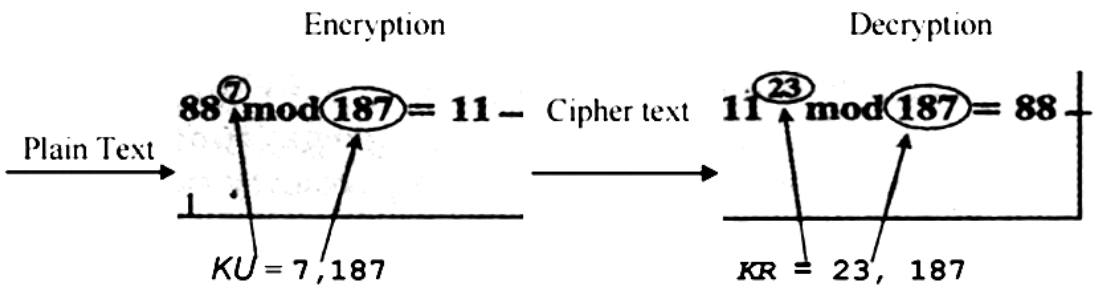
Select p, q		p ,q both prime $p \neq q$
Calculate $n = p \times q$		
Calculate $\phi(n) = (p - 1)(q - 1)$		
Select integer	e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate	d	$d = e^{-1} \bmod \phi(n)$
Public key		KU = { e,n}
Private key		KR = {d,n}

Encryption

Plaintext	$M < n$
Ciphertext	$C = M^e \pmod n$

Decryption

Ciphertext	C
Plaintext	$M = C^d \pmod n$



2. Security of RSA

There are three approaches to attack the RSA:

- brute force key search (infeasible given size of numbers)
- mathematical attacks (based on difficulty of computing $\phi(N)$, by factoring modulus
- timing attacks (on running time of decryption)

Factoring Problem

Mathematical approach takes 3 forms:

- ➢ Factor n = p*q, hence find $\phi(n)$ and then d.
- ➢ Determine $\phi(n)$directly without determining p and q and find d.
- ➢ Find d directly, without first determination $\phi(n)$.

3. Timing attacks

It has been proved that the opponent can determine a private key by keeping track of how long a computer takes to decipher messages. Although the timing attack is a serious threat, there are simple countermeasures that can be used:

- Constant exponentiation time – ensures that all exponentiations take the same amount of time before returning a result.
- Random delay – better performance could be achieved by adding a random delay to the exponentiation algorithm to confuse the timing attack.
- Blinding – multiply the ciphertext by a random number before performing exponentiation.

2.4 Key Management And Distribution

2.4.1 Symmetric Key Distribution Using Asymmetric Encryption

Q13. Explain about Symmetric Key distribution using Asymmetric Encryption.

Ans :

Because of the inefficiency of public key cryptosystems, they are almost never used for the direct encryption of sizable block of data, but are limited to relatively small blocks. One of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution.

Simple Secret Key Distribution

If A wishes to communicate with B, the following procedure is employed :

1. A generates a public/private key pair {PUa, PRa} and transmits a message to B
2. B generates a secret key, Ks, and transmits it to A, which is 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.

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.

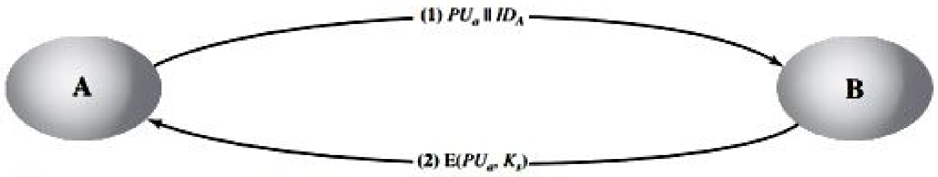


Figure Simple Use of Public-Key Encryption to Establish a Session Key

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 eaves dropping.

The protocol depicted in Figure is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message . Such an attack is knownas a man-in-the-middle attack. 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 beingdetected.

1. A generates a public/private key pair {PUa, PRa} and transmits a message intended for Bconsisting 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 butsimply eavesdrops. Knowing Ks, E can decrypt all messages, and both A and B are unaware of theproblem. Thus, this simple protocol is only useful in an environment where the only threat is eavesdropping.

Secret Key Distribution with Confidentiality and Authentication

Figure 14.8, 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 subsequently in this chapter. then the following steps occurs.

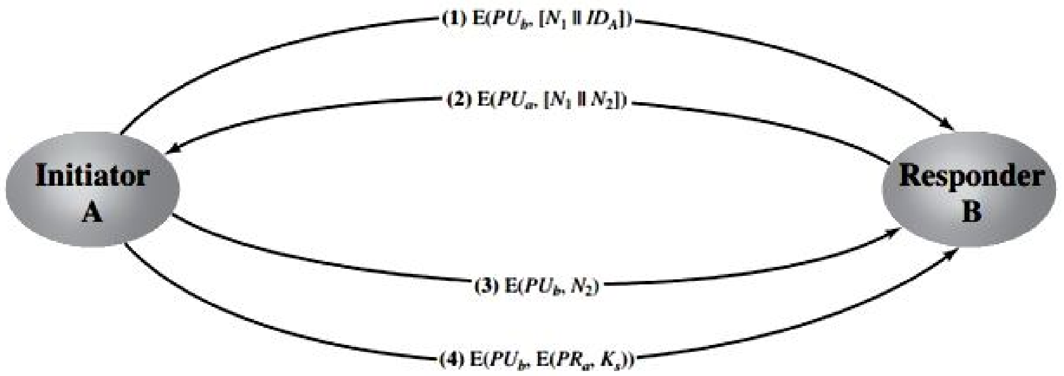

(1) E(PU_b, [N_1 || ID_A])
(2) E(PU_a, [N_1 || N_2])
Initiator A
Responder B
(3) E(PU_b, N_2)
(4) E(PU_b, E(PR_a, K_s))

Figure Public-Key Distribution of Secret Keys

1. A uses B's public key to encrypt a message to B containing an identifier of A(IDA) and a nonce (N1), which is used to identify this transaction uniquely.

   B sends a message to A encrypted with PUa and containing A's nonce (N1) as ell as a new nonce generated by B (N2). Because only B could have (N2). Because only B could have decrypted message (1), the presence of N1 in message (2) assures A that the correspondent is B.
2. A returns N2, encrypted using B's public key, to assure B that its correspondent is A.

   A selects a secret key Ks and sends M = E(PUb, E(PRa, Ks)) to B. Encryption of this message with B's publickey ensures that only B can read it; encryptionwith A's private key ensures that only A could have sent it.
3. B computes D(PUa, D(PRb, M)) to recover the secret key.
4. The result is that this scheme ensures both confidentiality and authentication in the exchange of a secret key.

2.4.2 Distribution of Public Keys

Q14. Explain about several techniques for distribution of public keys.

Ans :

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

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

Figure 14.9 Uncontrolled Public-Key Distribution

Public Announcement of Public Keys

The point of public-key encryption is that the public key is public. Thus, if there is somebroadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to anyother participant or broadcast the key to the community at large

Although this approach is convenient, it has a major weakness. Anyone can forger 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, theforger is able to read all encrypted messages intended for A and can use the forged keys for authentication.

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 14.10). 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 authenti- cated 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.

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

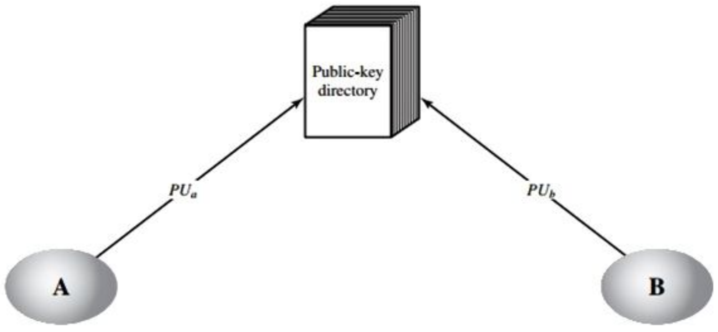


Figure 14.10 Public-Key Publication

Public-Key Authority

Stronger security for public-key distribution can be achieved by providing tighter control over thedistribution of public keys from the directory. 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 occur.

1. A sends a timestamped message to the public-key authority containing a request for thecurrent 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 themessage 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 used 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.

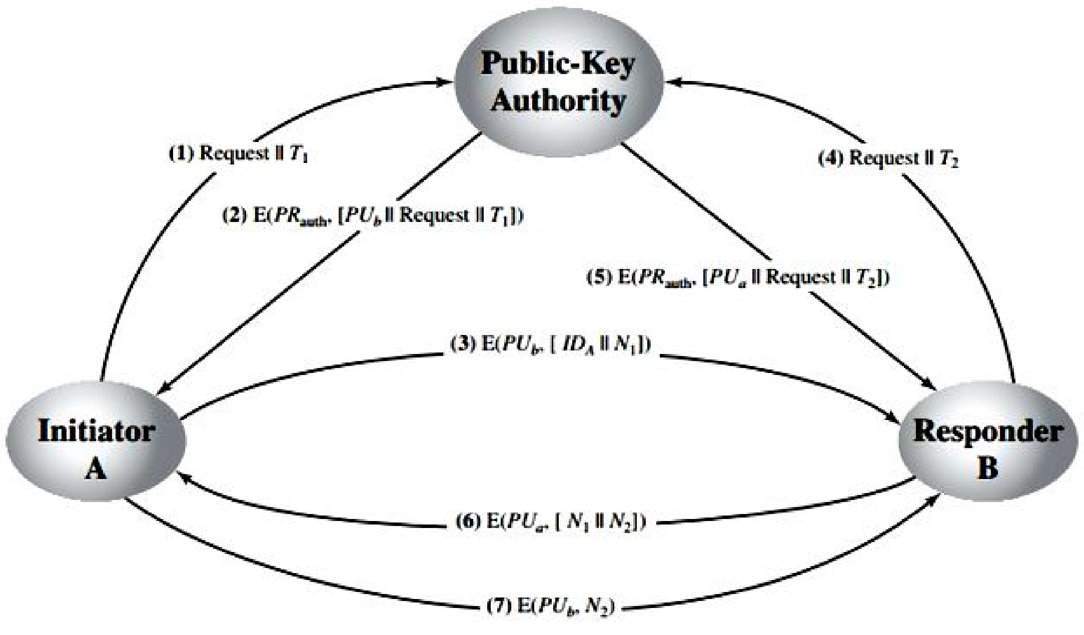

Public-Key Authority
(1) Request || T_1
(4) Request || T_2
(2) E(PR_{auth}, [PU_b || Request || T_1])
(5) E(PR_{auth}, [PU_a || Request || T_2])
(3) E(PU_b, [ID_A || N_1])
Initiator A
Responder B
(6) E(PU_a, [N_1 || N_2])
(7) E(PU_b, N_2)

Figure 14.11 Public-Key Distribution Scenario

The original timestamp given so A can determine that this is not an old mes- sage from the authoritycontaining 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 (N1), which is used to identify this transaction uniquely.

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

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 (N1) as well as a new nonce generated by B (N2). Because only B could have decrypted message (3), the presence of N1 in message (6) assures A that the correspondent is B.

6. A returns N2, which is encrypted using B's public key, to assure B that its cor- respondent is A.

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 ascaching. Periodically, a user should request fresh copies of the public keys of its correspondents to ensurecurrency.

Public-Key Certificates

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,, 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, an identifier of the key owner, and 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 needing 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 thecertificate 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'sowner.
2. Any participant can verify that the certificate originated from the certificate authority and is notcounterfeit.
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the currency of the certificate.

A certificate scheme is illustrated in Figure 14.12. Each participant applies to the certificate authority,supplying a public key and requesting a certificate.

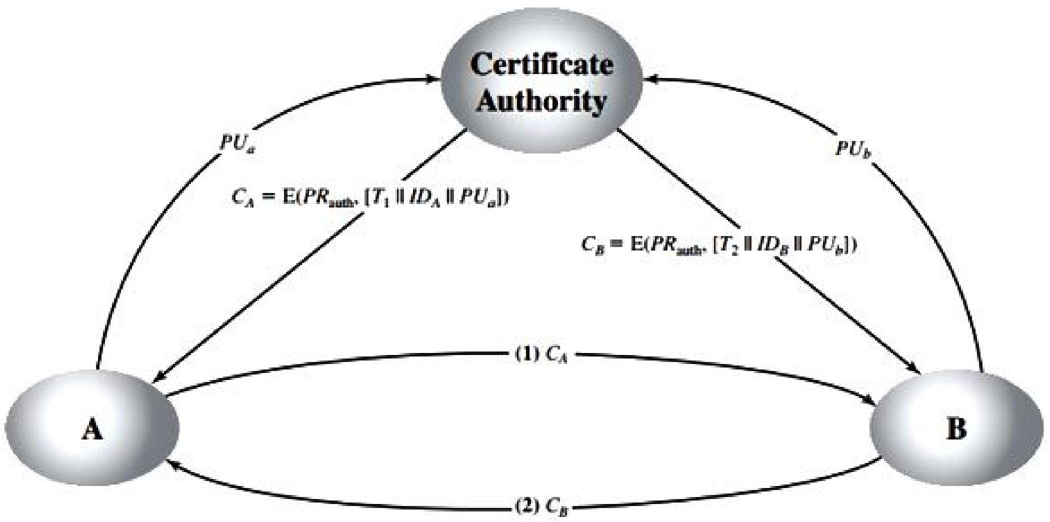

Certificate Authority
PU_a
PU_b
$C_A = E(PR_{auth}, [T_1 \| ID_A \| PU_a])$
$C_B = E(PR_{auth}, [T_2 \| ID_B \| PU_b])$
(1) C_A
(2) C_B
A
B

Figure 14.12 Exchange of Public-Key Certificates

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

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

where *PR*auth is the private key used by the authority and *T* is a timestamp. A may then pass this certificateon to any other participant, who reads and verifies the cer- tificate 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 time stamp 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 the compromised old public key, the adversary can read those messages.

2.4.3 X.509 Certificates

Q15. Write about x.509 certificates.

Ans :

ITU - t recommendation X.509 is part of the X.500 series of recommendations that define a directory service.

X.509 defines a framework for the provision of authentication services by theX.500 directory to its users. The directory may serve as a repository of public-key certificates of the type. Each certificate contains the public key of a user and is signed with the private keyof 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 inX.509 are used in a variety of contexts.

X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the useof a specific algorithm but recommends RSA.The dig- ital signature scheme is assumed to require the use of ahash function. Again, the stan- dard does not dictate a specific hash algorithm.

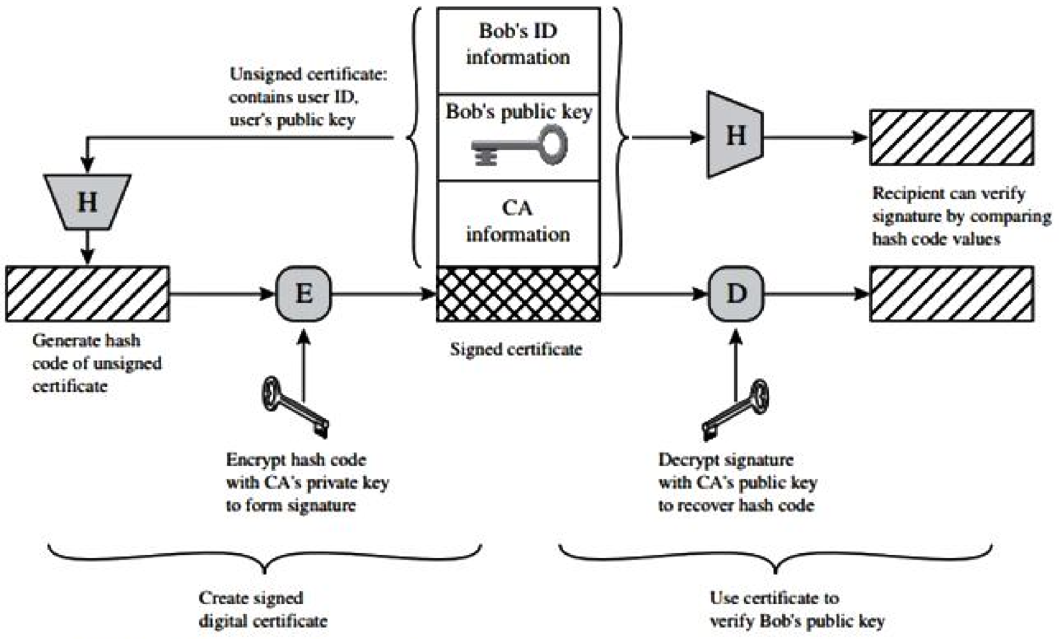

Bob's ID information
Bob's public key
CA information
Unsigned certificate: contains user ID, user's public key
H
E
D
Signed certificate
Recipient can verify signature by comparing hash code values
Generate hash code of unsigned certificate
Encrypt hash code with CA's private key to form signature
Decrypt signature with CA's public key to recover hash code
Create signed digital certificate
Use certificate to verify Bob's public key

Figure 14.13 Public-Key Certificate Use

Certificates

The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user. The directory server itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates. Figure 14.14a shows the general format of a certificate, which includes the following elements.

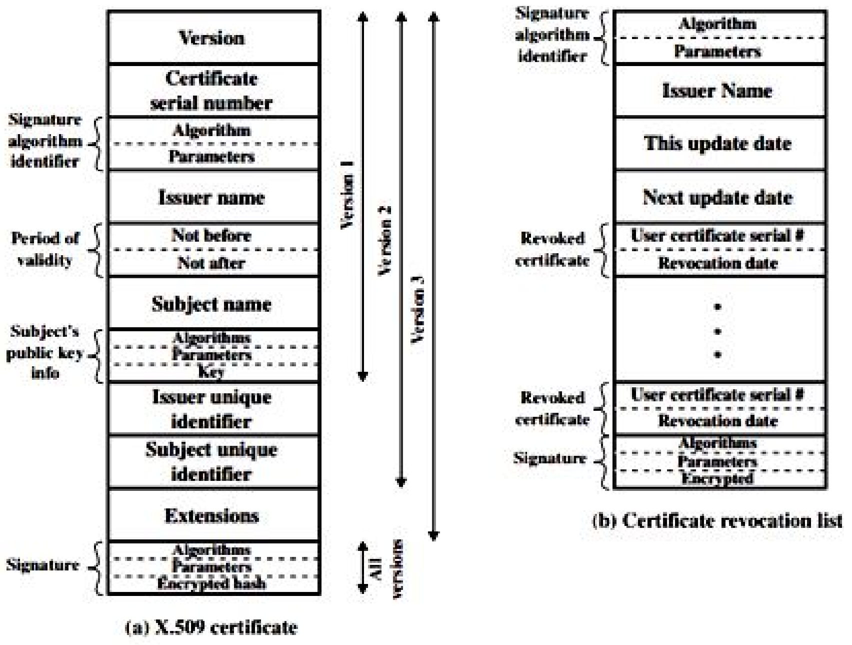


Figure 14.14 X.509 Formats

- **Version**

  Differentiates among successive versions of the certificate format; the default is version 1. If theissuer unique identifier or subject unique identifier are present, the value must be version 2. If one or moreextensions are present, the version must be version 3.

- **Serial Number**

  An integer value unique within the issuing CA that is unambiguously associated withthis certificate.

- **Signature Algorithm Identifier**

  The algorithm used to sign the certificate together with anyassociated parameters. Because this information is repeated in the signature field at the end of thecertificate, this field has little, if any, utility.

- **Issuer Name**

  X.500 is the name of the CA that created and signed this certificate.

- **Period of Validity**

  Consists of two dates: the first and last on which the certificate is valid.

- **Subject Name**

  The name of the user to whom this certificate refers. That is, this certificate certifies thepublic 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 forwhich 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 theevent 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 theevent 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 .

- **Signature**

  Covers all of the other fields of the certificate; it contains the hash code of the other fieldsencrypted with the CA's private key. This field includes the signature algorithm identifier.

The unique identifier fields were added in version 2 to handle the possible reuse of subject and/or issuer names over time. These fields are rarely used.

The standard uses the following notation to define a certificate:

CA << A >> = CA {V, SN, AI, CA, UCA, A, UA, Ap, TA}

where

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

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

V = version of the certificate

SN = serial number of the certificate

AI = identifier of the algorithm used to sign the certificate

CA = name of certificate authority

UCA = optional unique identifier of the CA

A = name of user A

UA = optional unique identifier of the user A

Ap = public key of user A

TA = period of validity of the certificate

The CA signs the certificate with its private key. If the corresponding public key is known to a user, then thatuser can verify that a certificate signed by the CA is valid. This is the typical digital signature approachillustrated in Figure 13.2.

2.4.4 The Diffie-Hellman Algorithm

Q16. Write about Diffie-Helman Algorithm

Ans :

This algorithm uses arithmetic modulus as the basis of its calculation. Suppose Alice and Bob follow this key exchange procedure with Eve acting as a man in middle interceptor .

Here are the calculation steps followed in this algorithm that make sure that eve never gets to know the final keys through which actual encryption of data takes place.

- First, both Alice and Bob agree upon a prime number and another number that has no factor in common. Lets call the prime number as **p** and the other number as **g**. Note that **g** is also known as the generator and **p** is known as prime modulus.
- Now, since eve is sitting in between and listening to this communication so eve also gets to know **p** and **g**.
- Now, the modulus arithmetic says that **r** = (**g** to the power **x**) mod **p.** So **r** will always produce an integer between 0 and **p**.
- The first trick here is that given **x** (with **g** and **p** known), its very easy to find r. But given **r** (with **g** and **p** known) its difficult to deduce **x**.
- One may argue that this is not that difficult to crack but what if the value of **p**is a very huge prime number? Well, if this is the case then deducing **x** (if **r** is given) becomes almost next to impossible as it would take thousands of years to crack this even with super computers.
- This is also called the discrete logarithmic problem.
- Coming back to the communication, all the three Bob, Alice and eve now know **g** and **p**.
- Now, Alice selects a random private number **xa** and calculates (**g** to the power **xa**) mod **p** = **ra**. This resultant **ra** is sent on the communication channel to Bob. Intercepting in between, eve also comes to know **ra**.
- Similarly Bob selects his own random private number **xb**, calculates (**g** to the power **xb**) mod **p** = **rb** and sends this **rb** to Alice through the same communication channel. Obviously eve also comes to know about **rb**.
- So eve now has information about **g**, **p**, **ra** and **rb**.
- Now comes the heart of this algorithm. Alice calculates (**rb** to the power **xa**) mod **p** = **Final key** which is equivalent to **(g to the power (xa*xb)) mod p** .
- Similarly Bob calculates **(ra to the power xb) mod p** = **Final key** which is again equivalent to **(g to the power(xb * xa)) mod p**.
- So both Alice and Bob were able to calculate a common **Final key** without sharing each others private random number and eve sitting in between will not be able to determine the **Final key** as the private numbers were never transferred.

Diffie-Hellman algorithm works perfectly to generate cryptographic keys which are used to encrypt the data being communicated over a public channel.

UNIT III

Cryptographic Hash Functions: Applications of Cryptographic Hash Functions, Two Simple Hash Functions, Secure Hash Algorithm (SHA) & MD5 Algorithm.Message Authentication Codes: Message Authentication Requirements, Message Authentication Functions, Requirements for Message Authentication Codes,Security of MACs,MACs Based on Hash Functions:HMAC,MACs Based on Block Ciphers:DAA and CMAC.Digital Signatures: Digital Signatures, NIST Digital Signatures Algorithm.

3.1 Cryptographic Hash Functions

3.1.1 Applications of Cryptographic Hash Functions

Q1. Write the applications of cryptographic has functions.

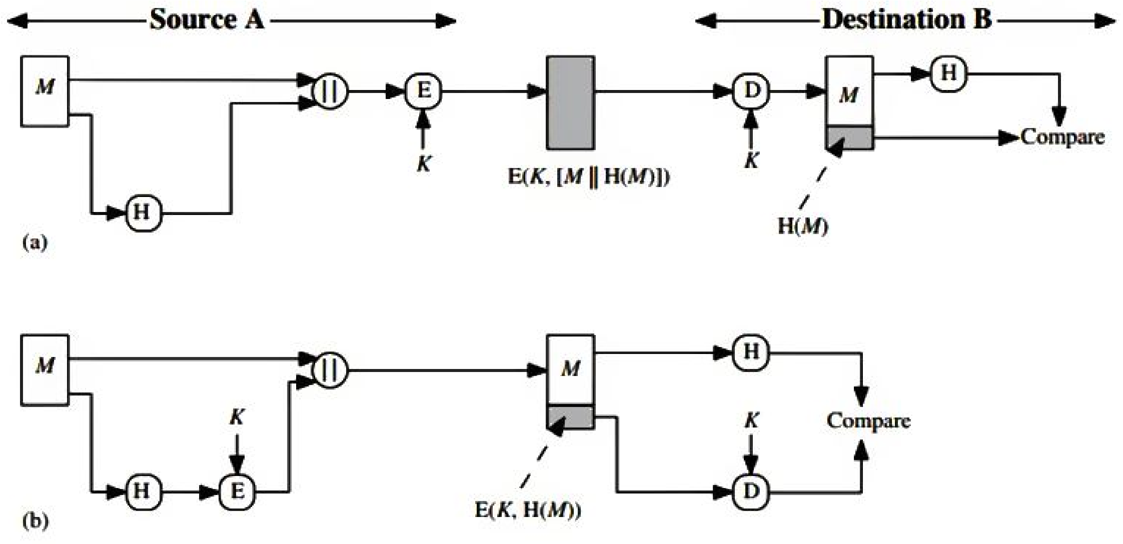
Ans :

Perhaps the most versatile cryptographic algorithm is the cryptographic hash function. It is used in a wide variety of security applications and Internet protocols.

1. Message Authentication

Message authentication is a mechanism or service used to verify the integrity of a message. Message authentication assures that data received are exactly as sent. In many cases, there is a requirement that the authentication mechanism assures that purported identity of the sender is valid. When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest.

Figure 1 illustrates a variety of ways in which a hash code can be used to provide message authentication, as follows.



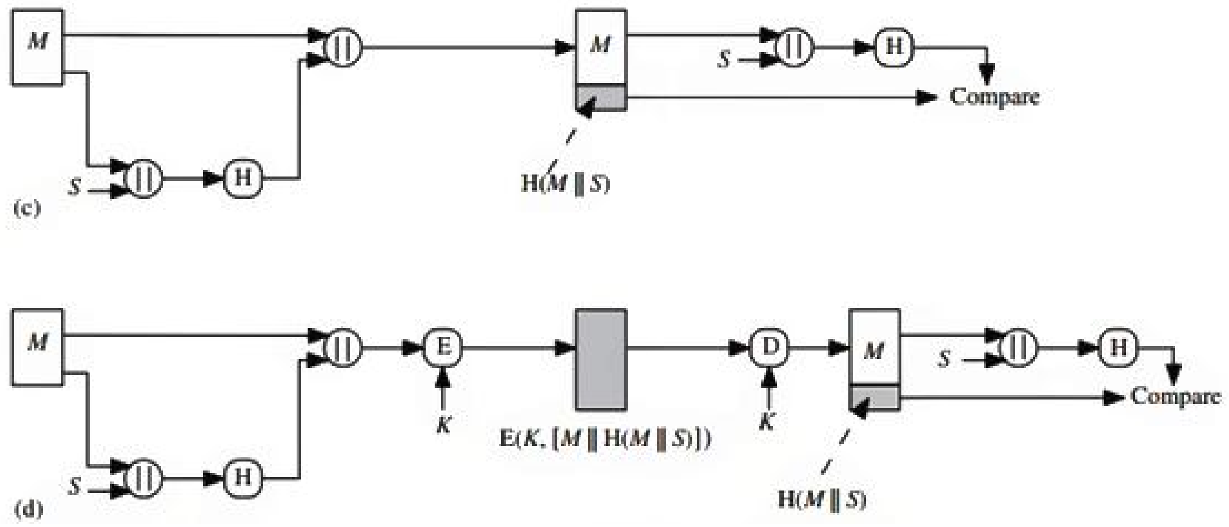


Figure 1 : Simplified Examples of the Use of a Hash Function for Message Authentication

a) The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.

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

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

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

When confidentiality is not required, method (b) has an advantage over methods. (a) and (d), which encrypts the entire message, in that less computation is required.

- ➢ Encryption software is relatively slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.
- ➢ Encryption hardware costs are not negligible. Low-cost chip implementations of DES are available, but the cost adds up if all nodes in a network must have this capability.
- ➢ Encryption hardware is optimized toward large data sizes. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead.
- ➢ Encryption algorithms may be covered by patents, and there is a cost associated with licensing their use.

More commonly, message authentication is achieved using a message authentication code (MAC), also known as a keyed hash function.

Typically, MACs are used between two parties that share a secret key to authenticate information exchanged between those parties.

2. Digital Signatures

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

Figure 2 illustrates, in a simplified fashion, how a hash code is used to provide a digital signature.

a) The hash code is encrypted, using public-key encryption with the sender's private key.

b) f confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key. This is a common technique.

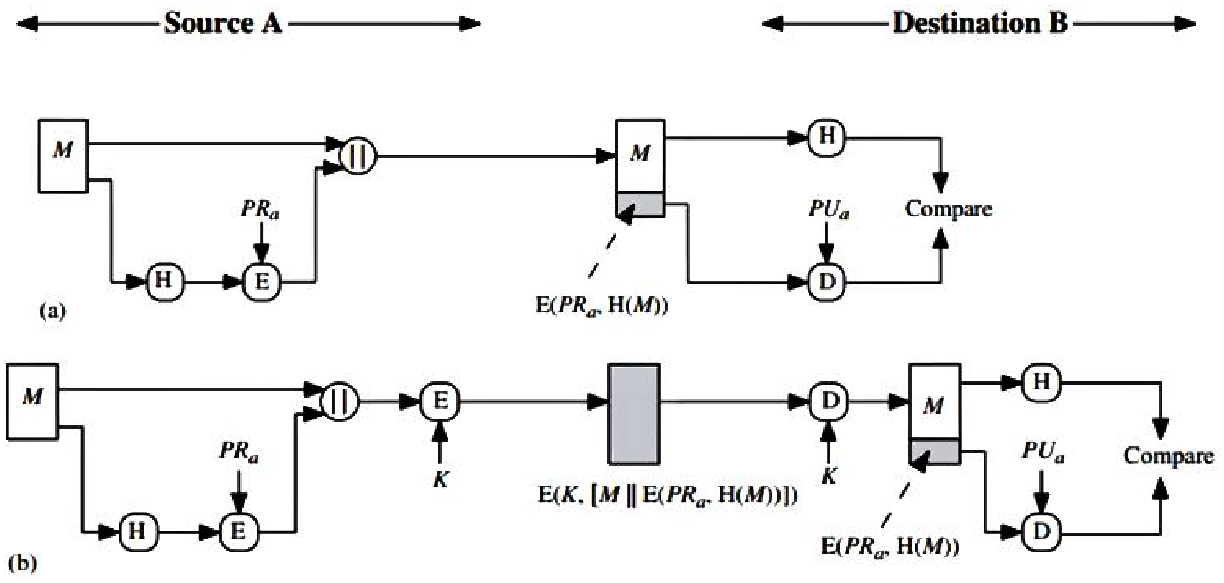


Figure 2 : Simplied Examples of Digital Signature

3. Other Applications

Hash functions are commonly used to create a one-way password file. Thus, the actual password is not retrievable by a hacker who gains access to the password file. In simple terms, when a user enters a password, the hash of that password is compared to the stored hash value for verification. This approach to password protection is used by most operating systems.

Hash functions can be used for intrusion detection and virus detection. Store H(F) for each file on a system and secure the hash values. One can later determine if a file has been modified by re-computing H(F). An intruder would need to change F without changing H(F).

3.2 Two Simple Hash Functions

Q2. Write about two simple hash functions.

Ans :

To get some feel for the security considerations involved in cryptographic hash functions, we present two simple, insecure hash functions in this section. 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

$$C_i = b_{i1} \oplus b_{i2} \oplus .... \oplus b_{im}$$

Where,

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

m = number of n-bit block in the input

b_{ij} = ith bit in jth block

$\oplus$ = XOR operation

This operation produces a simple parity for each bit position and is known as a longitudinal redundancy check. It is reasonably effective for random data as a data integrity check. Each n-bit hash value is equally likely. 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.

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.

This has the effect of "randomizing" the input more completely and overcoming any regularities that appear in the input. Figure 3 illustrates these two types of hash functions for 16-bit hash values.

Although the second procedure provides a good measure of data integrity, it is virtually useless for data security when an encrypted hash code is used with a plain text message.

Although a simple XOR or rotated XOR (RXOR) is insufficient if only the hash code is encrypted, you may still feel that such a simple function could be useful when the message together with the hash code is encrypted. We can define the scheme as follows: Given a message M consisting of a sequence of 64-bit blocks X1, X2, Á , XN, define the hash code h = H(M) as the block-by-block XOR of all blocks and append the hash code as the final block:

$$h = X_{N+1} = X1 \otimes X2 \otimes Á \otimes X_N$$

Next, encrypt the entire message plus hash code using CBC mode to produce the encrypted message $Y_1, Y_2, ..., Y_{N+1}$.

$$X_1 = IV \oplus D(K, Y_1)$$

$$X_i = Y_{i-1} \oplus D(K, Y_i)$$

$$X_{N+1} = Y_N \oplus D(K, Y_{N+1})$$

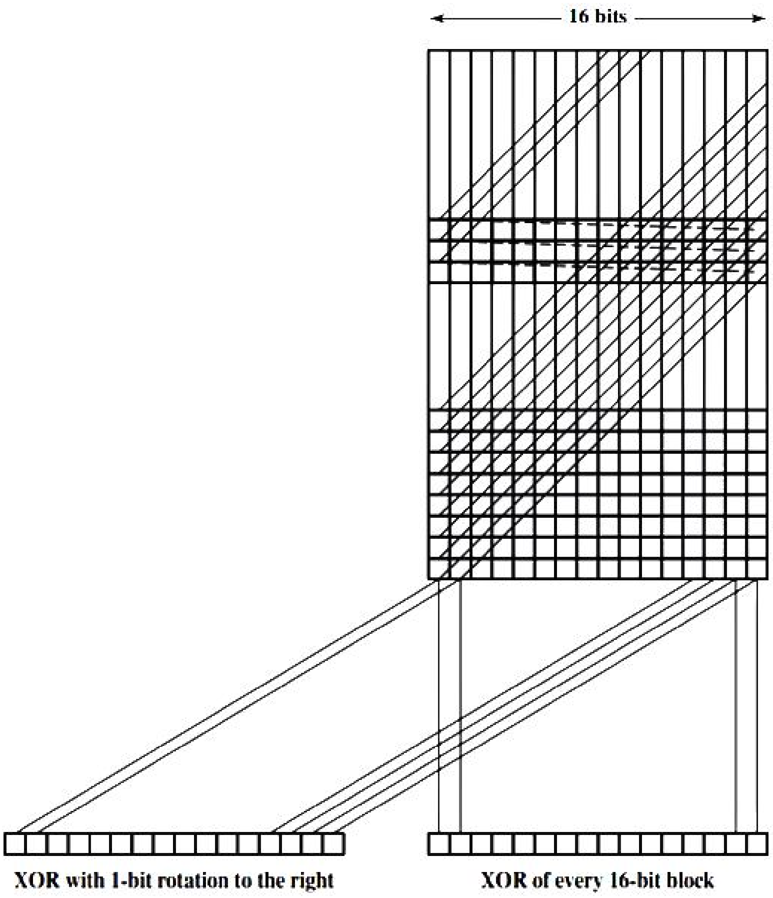

16 bits
XOR with 1-bit rotation to the right
XOR of every 16-bit block

Figure 3 : Two Simple Hash Function

But, X_{N+1} = is the hash code:

$$X_{N+1} = X_1 \oplus X_2 \oplus \ldots \oplus X_N$$

$$= IV \oplus D(K, Y_1)] \oplus [Y_1 \oplus D(K, Y_2)] \oplus \cdots \oplus [Y_{N-1} \oplus D(K, Y_N)]$$

Because the terms in the preceding equation can be XORed in any order, it follows that the hash code would not change if the cip her text blocks were permuted.

3.3 Secure Hash Algorithm (SHA)

Q4. What is secure hash algorithm? Explain.

Ans :

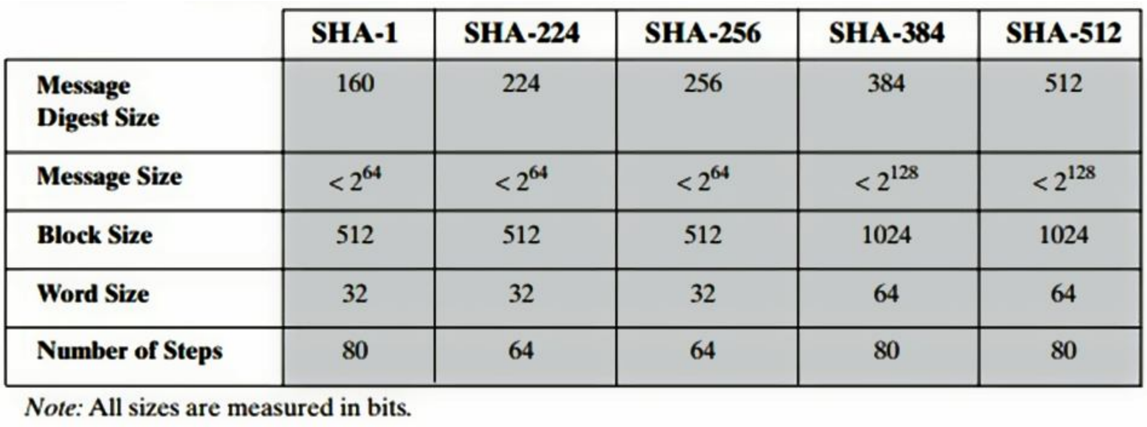
The most widely used hash function has been the Secure Hash Algorithm (SHA). When weaknesses were discovered in SHA, now known as SHA-0, a revised version was issued as FIPS 180-1 in 1995 and is referred to as SHA-1. The actual standards document is entitled “Secure Hash Standard.” SHA is based on the hash function MD4, and its design closely models MD4. SHA-1 is also specified in RFC 3174, which essentially duplicates the material in FIPS 180-1 but adds a C code implementation.

SHA-1 produces a hash value of 160 bits that defined three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known asSHA-256, SHA-384, and SHA-512, respectively. Collectively, these hash algorithms are known as SHA-2. These new versions have the same underlying structure and use the same types of modular arithmetic and logical binary operations as SHA-1. A revised document was issued as FIP PUB 180-3 in 2008, which added a 224-bit version (Table). SHA-2 is also specified in RFC 4634, which essentially duplicates the material in FIPS 180-3 but adds a C code implementation.

SHA-512 Logic

The algorithm takes as input a message with a maximum length of less than 2128 bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks. Figure 4 depicts the overall processing of a message to produce a digest. The processing consists of the following steps.

Table : Comparison of SHA Parameters



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

Note: All sizes are measured in bits.

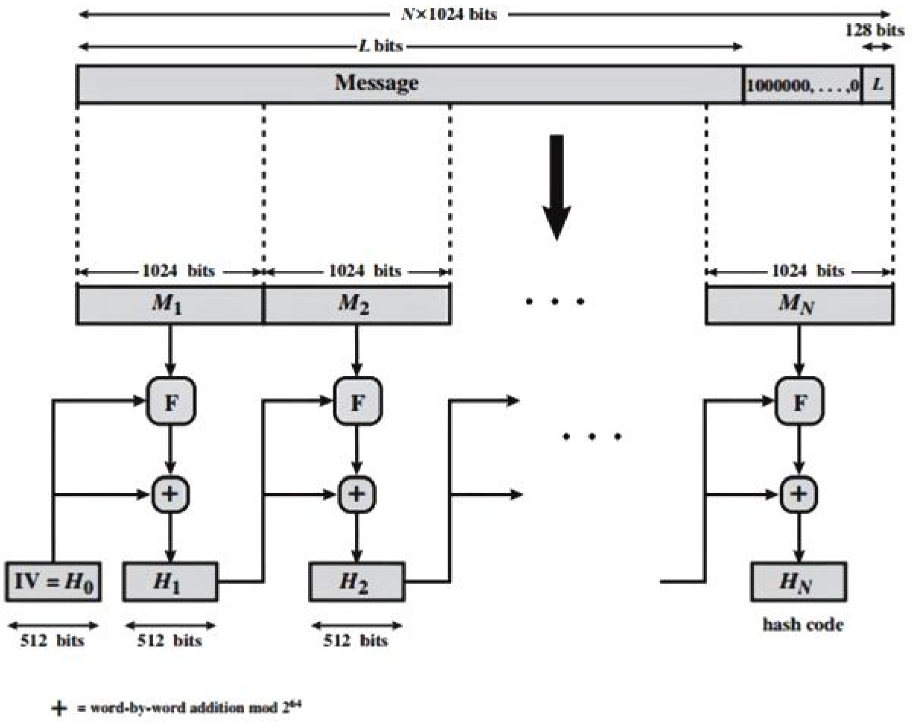


Figure 4 : Message Digest Generation Using SHA-512

Step 1 : Append padding bits

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

Step 2 : Append length

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

The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. In Figure 4, the expanded message is represented as the sequence of 1024-bit blocks *M*1, *M*2, ... , *MN*, so that the total length of the expanded message is *N* * 1024 bits.

Step 3 : Initialize hash buffer

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

a = 6A09E667F3BCC908

e = 510E527FADE682D1

b = BB67AE8584CAA73B

f = 9B05688C2B3E6C1F

c = 3C6EF372FE94F82B

g = 1F83D9ABFB41BD6B

d = A54FF53A5F1D36F1

h = 5BE0CD19137E2179

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

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

The heart of the algorithm is a module that consists of 80 rounds; this module is labeled Fin Figure 4. The logic is illustrated in Figure 5.

Each round takes as input the 512-bit buffer value, abcdefgh, and updates the contents of the buffer.

At input to the first round, the buffer has the value of the intermediate hash value, Hi - 1. Each round t makes use of a 64-bit value Wt, derived from the current 1024-bit block being processed (Mi). These values are derived using a message schedule described subsequently. Each round also makes use of an additive constant Kt, where 0 … t … 79 indicates one of the 80 rounds. These words represent the first 64 bits of the fractional parts of the cube roots of the first 80 prime numbers.

The output of the eightieth round is added to the input to the first round (Hi - 1) to produce Hi. The addition is done independently for each of the eight words in the buffer with each of the corresponding words in *Hi*-1, using addition modulo 264.

Step 5 : Output

After all *N* 1024-bit blocks have been processed, the output from the *N*th stage is the 512-bit message digest.

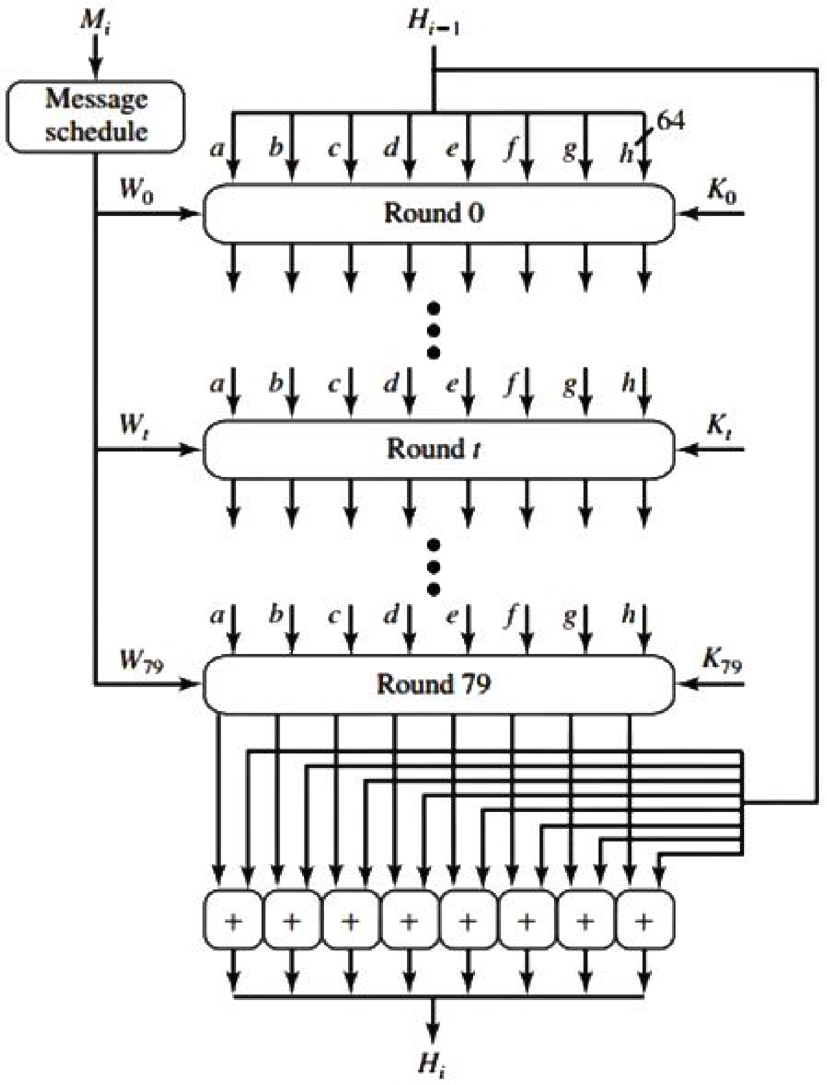


Figure 5 : SHA-512 Processing of a Single 1024-Bit Block

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

H_0 = IV

H_i = SUM64(H_{i-1}, abcdefghi)

MD = H_N

Where,

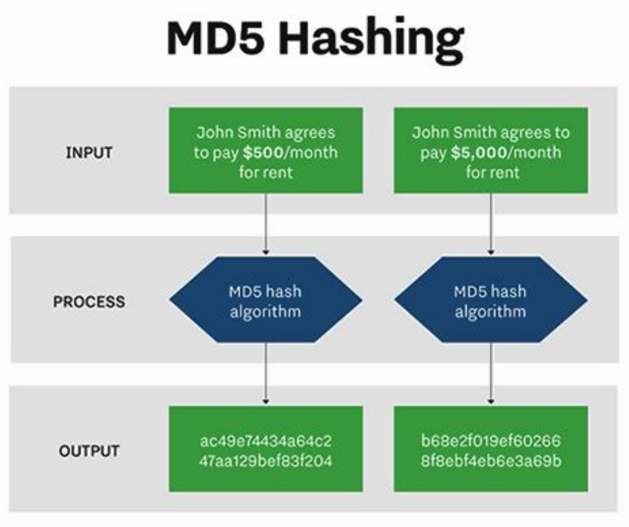
IV	=	Initial value of the abcdefgh buffer, defined in step 3.
abcdefghi	=	The output of the last round of processing of the ith message block.
N	=	The number of blocks in the message (including padding and length fields).
SUM_{64}	=	Addition modulo 2^{64} performed separately on each word of the pair of inputs.
MD	=	Final message digest value.

3.4 MD5

Q5. Explain about MD5 algorithm.

Ans :

The MD5 hashing algorithm is a one-way cryptographic function that accepts a message of any length as input and returns as output a fixed-length digest value to be used for authenticating the original message. The MD5 hash function was originally designed for use as a secure cryptographic hash algorithm for authenticating digital signatures. MD5 has been deprecated for uses other than as a non-cryptographic checksum to verify data integrity and detect unintentional data corruption.


MD5 Hashing
INPUT
John Smith agrees to pay $500/month for rent
John Smith agrees to pay $5,000/month for rent
PROCESS
MD5 hash algorithm
MD5 hash algorithm
OUTPUT
ac49e74434a64c2 47aa129bef83f204
b68e2f019ef60266 8f8ebf4eb6e3a69b

Message Digest Algorithm Characteristics

Message digests, also known as hash functions, are one-way functions; they accept a message of any size as input, and produce as output a fixed-length message digest.

MD5 is the third message digest algorithm created by Rivest.

How MD5 works

The MD5 message digest hashing algorithm processes data in 512-bit blocks, broken down into 16 words composed of 32 bits each. The output from MD5 is a 128-bit message digest value.

Computation of the MD5 digest value is performed in separate stages that process each 512-bit block of data along with the value computed in the preceding stage. The first stage begins with the message digest values initialized using consecutive hexadecimal numerical values. Each stage includes four message digest passes which manipulate values in the current data block and values processed from the previous block. The final value computed from the last block becomes the MD5 digest for that block.

MD5 Security

The goal of any message digest function is to produce digests that appear to be random. To be considered cryptographically secure, the hash function should meet two requirements: first, that it is impossible for an attacker to generate a message matching a specific hash value; and second, that it is impossible for an attacker to create two messages that produce the same hash value.

MD5 hashes are no longer considered cryptographically secure, and they should not be used for cryptographic authentication.

3.5 Message Authentication Codes

3.5.1 Message Authentication Requirements

Q6. What are the message authentication requirements?

Ans :

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

1. **Disclosure:** Release of message contents to any person or process not possessing the appropriate cryptographic key.
2. **Traffic analysis:** Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connection less environment, the number and length of messages between parties could be determined.
3. **Masquerade:** Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or nonreceipt by someone other than the message recipient.
4. **Content modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.
5. **Sequence modification:** Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
6. **Timing modification:** Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connection less application, an individual message (e.g.,data- gram) could be delayed or replayed.
7. **Source repudiation:** Denial of transmission of message by source.
8. **Destination repudiation:** Denial of receipt of message by destination.

Measures to deal with the first two attacks are in the realm of message confidentiality and are dealt with in Part One. Measures to deal with items (3) through (2) in the foregoing list are generally regarded as message authentication. Mechanisms for dealing specifically with item (7) come under the heading of digital signatures. Generally, a digital signature technique will also counter some or all of the attacks listed under items (3) through (6). Dealing with item (8) may require a combination of the use of digital signatures and a protocol designed to counter this attack.

In summary, message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness. A digital signature is an authentication technique that also includes measures to counter repudiation by the source.

3.5.2 Message Authentication Functions

Q7. Explain about message authentication functions.

Ans :

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

These may be grouped into three classes.

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

Message Encryption

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

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

Given a decryption function D and a secret key K, the destination will accept any input X and produce output Y = D(K, X). If X is the ciphertext of a legitimate message M produced by the corresponding encryption function, then Y is some plain text message M. Otherwise, Y willlikely be a meaningless sequence of bits.

The implications of the line of reasoning in the preceding paragraph are profound from the point of view of authentication. Suppose the message *M* can be any arbitrary bit pattern. In that case, there is no way to determine automatically, at the destination, whether an incoming message is the ciphertext of a legitimate message. This conclusion is incontrovertible: If *M* can be any bit pattern, then regardless of the value of *X*, the value *Y* = D(*K*, *X*) is *some* bit pattern and therefore must be accepted as authentic plain text.

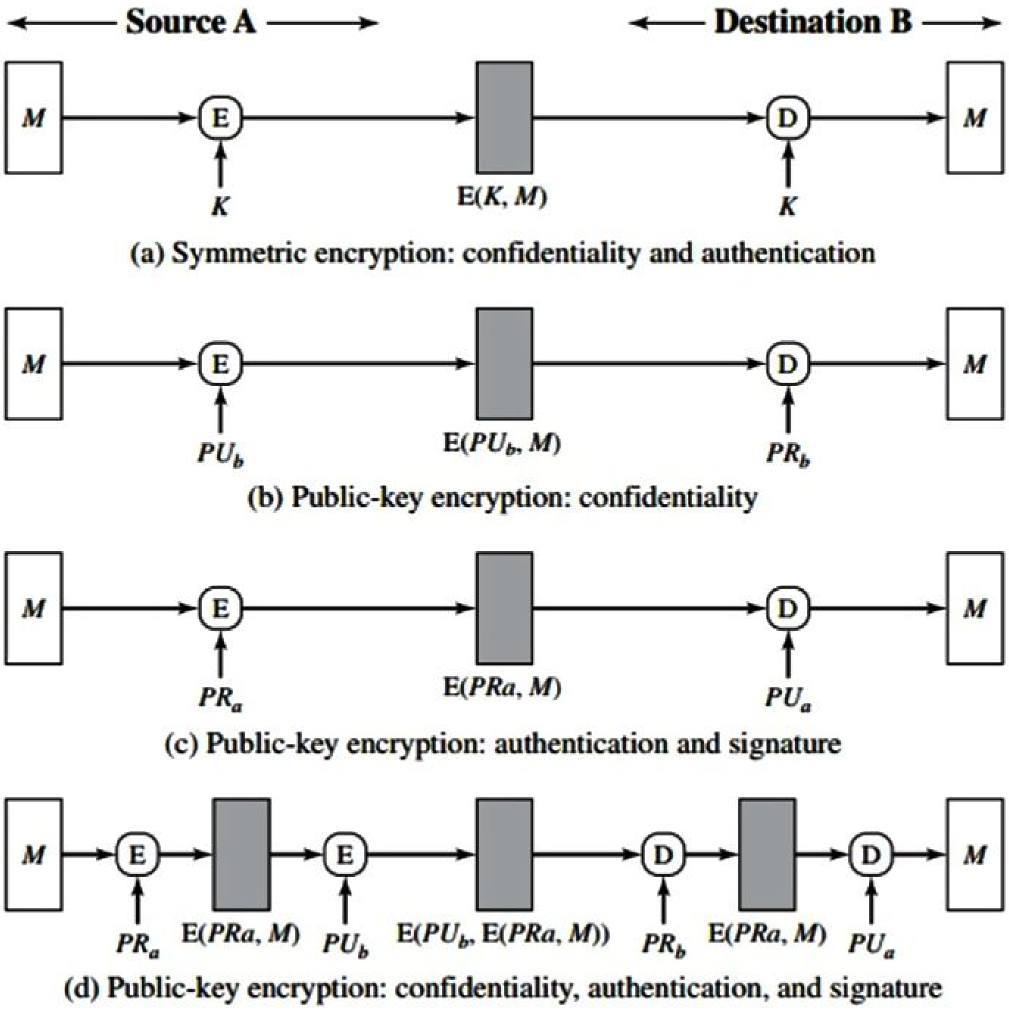

Source A
Destination B
M
E
D
M
K
E(K, M)
K
(a) Symmetric encryption: confidentiality and authentication
PU_b
$E(PU_b, M)$
PR_b
(b) Public-key encryption: confidentiality
PR_a
E(PRa, M)
PU_a
(c) Public-key encryption: authentication and signature
PR_a
E(PRa, M)
PU_b
$E(PU_b, E(PRa, M))$
PR_b
E(PRa, M)
PU_a
(d) Public-key encryption: confidentiality, authentication, and signature

Figure : Basic Uses of Message Encryption

Thus, in general, we require that only a small subset of all possible bit pat- terns be considered legitimate plain text. In that case, any spurious ciphertext is unlikely to produce legitimate plain text.

For a number of applications and encryption schemes, the desired conditions prevail as a matter of course. For example, suppose that we are transmitting English- language messages using a Caesar cipher with a shift of one ($K = 1$). A sends the following legitimate ciphertext:

Nbsftfbupbutboeepftfbupbutboemjuumfmbnctfbujwz

B decrypts to produce the following plain text: maresea to ats and doese a to ats and little lambseativy A simple frequency analysis confirms that this message has the profile of ordinary English. On the other hand, if an opponent generates the following random sequence of letters:

zuvrsoevgqxlzwigamdvnmhpmccxiuureosfbcebtqxsxq this decrypts to

ytuqrndufpwkyvhfzlcumlgolbbwhttqdnreabdaspwrwp

which does not fit the profile of ordinary English.

One solution to this problem is to force the plain text to have some structure that is easily recognized but that cannot be replicated without recourse to the encryption function.

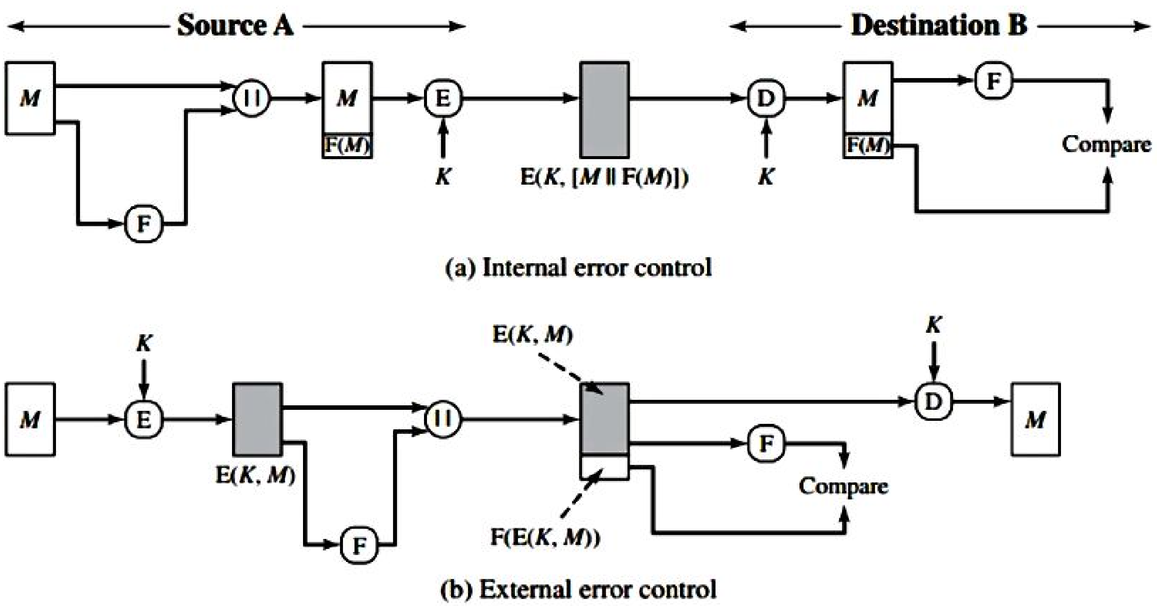


Figure : Internal and External Error Control

Message Authentication Code

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

$$\text{MAC} = \text{MAC}(K, M)$$

Where,

M = input message

C = MAC function

K = shared secret key

MAC = message authentication code

The message plus MAC are transmitted to the intended recipient. The recipient per- forms 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). 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 , 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 be 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 2n possible MACs, whereas there are N possible messages with N 77 2n. Furthermore, with a k-bit key, there are 2k possible keys.

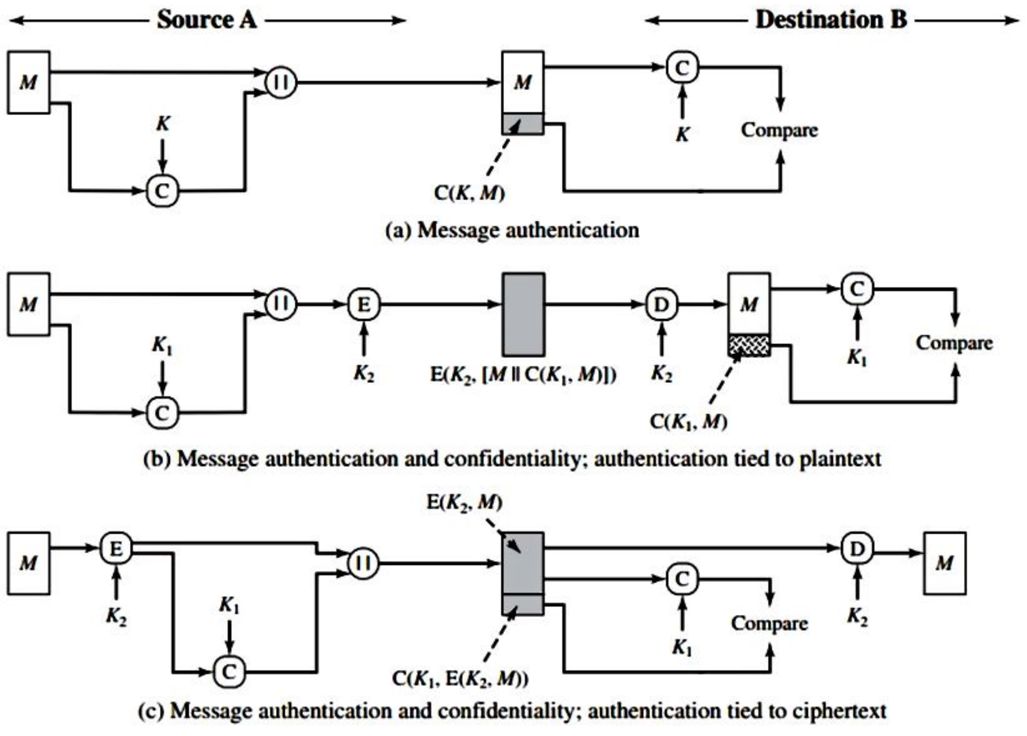

Source A
Destination B
M
K
C
||
Compare
C(K, M)
(a) Message authentication
K1
K2
E
D
E(K2, [M || C(K1, M)])
C(K1, M)
(b) Message authentication and confidentiality; authentication tied to plaintext
E(K2, M)
C(K1, E(K2, M))
(c) Message authentication and confidentiality; authentication tied to ciphertext

Figure : Basic Uses of Message Authentication Code (MAC)

Three situations in which a message authentication code is used.

1. There are a number of applications in which the same message is broadcast to a number of destinations. the message must be broadcast in plain text with an associated message authentication code. The responsible system has the secret key and performs authentication. If a violation occurs, the other destination systems are alerted by a general alarm.

2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis, messages being chosen at random for checking.
3. Authentication of a computer program in plain text is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources. However, if a message authentication code were attached to the program, it could be checked whenever assurance was required of the integrity of the program. Three other rationales may be added.
4. Separation of authentication and confidentiality functions affords architectural flexibility.
5. A user may wish to prolong the period of protection beyond the time of reception and yet allow processing of message contents. With message encryption, the protection is lost when the message is decrypted, so the message is protected against fraudulent modifications only in transit but not within the target system.
6. Finally, note that the MAC does not provide a digital signature, because both sender and receiver share the same key.

3.5.3 Requirements For Message Authentication Codes

Q8. What are the requirements for message authentication codes? Explain briefly.

Ans :

A MAC, also known as a cryptographic checksum, is generated by a function C of the form.

$$T = \text{MAC}(K, M)$$

where M is a variable-length message, K is a secret key shared only by sender and receiver, and MAC(K, M) is the fixed-length authenticator, sometimes called a tag. The tag 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 tag.

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 particular, for a ciphertext- only attack, the opponent, given ciphertext C, performs Pi = D(Ki, C) for all possible key values Ki until a Pi is produced that matches the form of acceptable plain text.

Suppose k 7 n; that is, suppose that the key size is greater than the MAC size. Then, given a known M1 and T_1, with T_1 = MAC(K, M_1), the cryptanalyst can perform Ti = MAC(K_i, M_1) for all possible key values ki. At least one key is guaranteed to produce a match of $T_i = T_1$. Note that a total of 2k tags will be produced, but there are only 2n 6 2k different tag values. Thus, a number of keys will produce the correct tag and the opponent has no way of knowing which is the correct key. On average, a total of 2k/2n = 2(k - n) keys will produce a match. Thus, the opponent must iterate the attack.

➢ **Round 1**

Given : M_1, T_1 = MAC(K, M_1)

Compute T_i = MAC(K_i, M_1) for all 2^k keys

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

➢ **Round 2**

Given M_2, T_2 = MAC (K, M_2)

Compute T_i = MAC(K_i, M_2)for the $2^{(k-n)}$ keys resulting from round 1 number of matches ≈ $2^{(k-2\times n)}$

And so on. On average, a rounds will be needed if $k = a \times n$. For example, if an 80-bit key is used and the tab is 32 bits, then the first round will produce about 248 possible keys. The second round will narrow the possible keys to about 216 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 tag length, then it is likely that a first round will produce a single match. It is possible that more than one key will produce such a match, in which case the opponent would need to perform the same test on a new (message, tag) pair.

Consider the following MAC algorithm. Let $M = (X1||X2||Á||Xm)$ be a message that is treated as a concatenation of 64-bit blocks Xi. Then define.

$$\Delta(M) = X_1 \oplus X_2 \oplus \ldots \oplus X_m$$

$$MAC(K, M) = E(K, \Delta(M))$$

where Í$ is the exclusive-OR (XOR) operation and the encryption algorithm is DES in electronic code book mode. Thus, the key length is 56 bits, and the tag length is 64 bits. If an opponent observes {M || MAC(K, M)}, a brute-force attempt to determine K will require at least 256 encryptions. But the opponent can attack the system by replacing X_1 through X_{m-1} with any desired values Y1 through Ym - 1and replacing Xm with Y_m, where Y_m is calculated as

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

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

Then the MAC function should satisfy the following requirements.

1. If an opponent observes M and MAC(K, M), it should be computationally infeasible for the opponent to construct a message M¿ such that

   MAC(K,'') = MAC(K, M)

2. MAC(K, M) should be uniformly distributed in the sense that for randomly choosen messages, M and M¿, the probability that MAC(K, M) = MAC(K, M') is 2 - n, where n is the number of bits in the tag.

3. Let M''' be equal to some known transformation on M. That is, M'' = f(M). For example, f may involve inverting one or more specific bits. In that case,

   Pr[MAC(K, M) = MAC(K, M')] = 2 – n

3.5.4 Security Of Macs

Q9. Write about various attacks on MACs.

Ans :

We can group attacks on MACs into two categories: brute-force attacks and cryptanalysis.

Brute-Force Attacks

A brute-force attack on a MAC is a more difficult undertaking than a brute-force attack on a hash function because it requires known message-tag pairs. 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 offline. Whether an off-line attack can be used on a MAC algorithm depends on the relative size of the key and the tag.

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 [xi, MAC(K, xi)], it is computationally infeasible to compute any text-MAC pair [x, MAC(K, x)] for any new input x|= xi.

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.

If an attacker can determine the MAC key, then it is possible to generate a valid MAC value for any input x. Suppose the key size is k bits and that the attacker has one known text–tag pair. Then the attacker can compute the n-bit tag on the known text for all possible keys. At least one key is guaranteed to produce the correct tag, namely, the valid key that was initially used to produce the known text–tag pair.

Cryptanalysis

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

3.6 MACs Based On Hash Functions: HMAC

Q10. Explain about HMAC algorithm.

Ans :

HMAC Design Objectives

RFC 2104 lists the following design objectives for HMAC.

- ➢ To use, without modifications, available hash functions. In particular, to use hash functions that perform well in software and for which code is freely and widely available.
- ➢ To allow for easy replace ability of the embedded hash function in case faster or more secure hash functions are found or required.
- ➢ To preserve the original performance of the hash function without incurring a significant degradation.
- ➢ To use and handle keys in a simple way.
- ➢ To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions about the embedded hash function.

HMAC Algorithm

Figure illustrates the overall operation of HMAC.

Define the following terms.

H	=	embedded hash function
IV	=	initial value input to hash function
M	=	message input to HMAC
Y_i	=	i th block of M,0 Š i Š (L – 1)
L	=	number of blocks in M
b	=	number of bits in a block
n	=	length of hash code produced by embedded hash function
K	=	secret key; recommended length is Š *n*; if key length is greater than *b*, the key is input to the hash function to produce an *n*-bit key

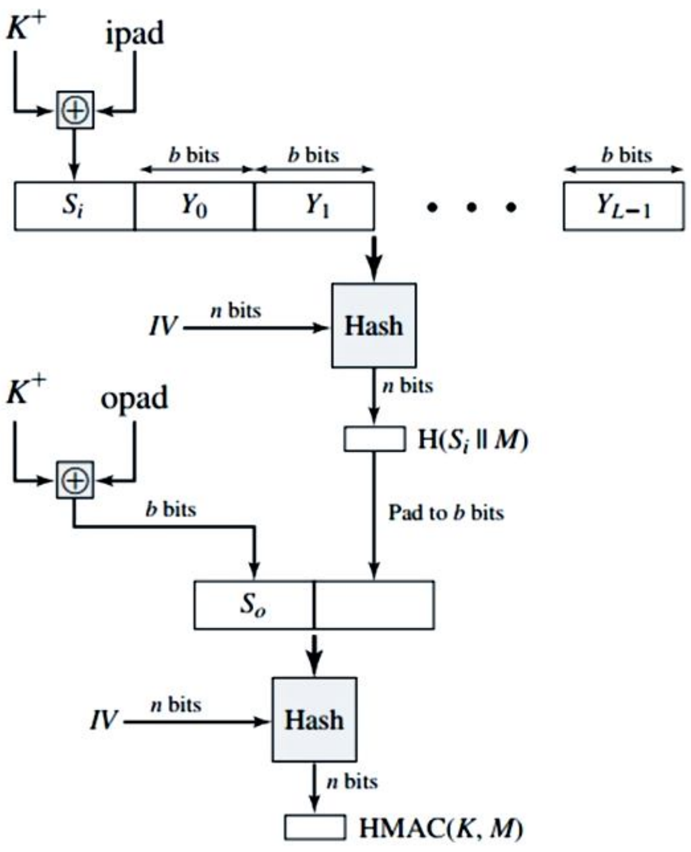


Figure : HMAC Structure

K+ = K padded with zeros on the left so that the result is b bits in length ipad = 00110110 (36 in hexadecimal) repeated b/8 times opad = 01011100 (5C in hexadecimal) repeated b/8 times Then HMAC can be expressed as HMAC(K, M) = H[(K+ $\otimes$ opad) || H[(K+ $\otimes$ ipad) || M]]

We can describe the algorithm as follows.

1. Append zeros to the left end of K to create a b-bit string K+ (e.g., if K is of length 160 bits and b= 512, then K will be appended with 44 zeroes).
2. XOR (bitwise exclusive-OR) K+ with ipad to produce the b-bit block Si.
3. Append M to Si.
4. Apply H to the stream generated in step 3.
5. XOR K+ with opad to produce the b-bit block So.
6. Append the hash result from step 4 to So.
7. Apply H to the stream generated in step 6 and output the result.

A more efficient implementation is possible, as shown in Figure.

Two quantities are precomputed:

f(IV, (K + $\otimes$ ipad))

f(IV, (K + $\otimes$ opad))

where f(cv, block) is the compression function for the hash function,

Security of HMAC

The security of any MAC function based on an embedded hash function depends in some way on the cryptographic strength of the underlying hash function.

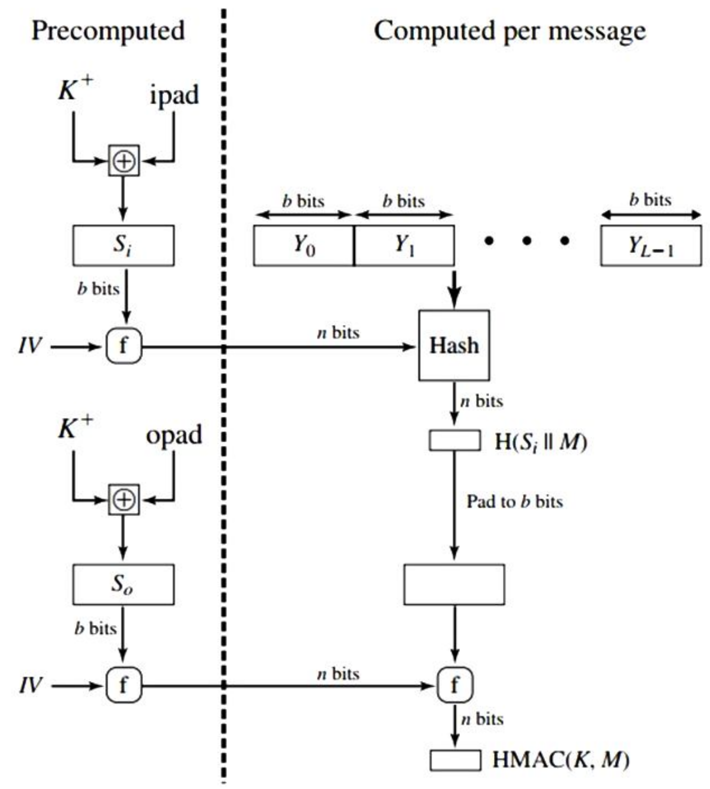


Figure : Efficient Implementation of HMAC

The security of a MAC function is generally expressed in terms of the probability of successful forgery with a given amount of time spent by the forger and a given number of message–tag pairs created with the same key. the probability of successful attack on HMAC is equivalent to one of the following attacks on the embedded hash function.

1. The attacker is able to compute an output of the compression function even with an IV that is random, secret, and unknown to the attacker.
2. The attacker finds collisions in the hash function even when the IV is random and secret.

3.7 MACs Based On Block Ciphers: DAA And CMAC

Q11. Explain about Data Authentication algorithm (DAA).

Ans :

Data Authentication Algorithm

The **Data Authentication Algorithm** (DAA), based on DES, has been one of the most widely used MACsfor a number of years.

The algorithm can be defined as using the cipher block chaining (CBC) mode of operation of DES (Figure) with an initialization vector of zero. The data (e.g., message, record, file, or program) 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 zeroes to form a full 64-bit block. Using the DES encryption algorithm E and a secret key K, a data authentication code (DAC) is calculated as follows (Figure).

$O_1 = E(K, D)$

$O_2 = E(K, [D_2 \oplus O_1])$

$O_3 = E(K, [D_3 \oplus O_2])$

$\vdots$

$O_N = E(K, [D_N \oplus O_{N-1}])$

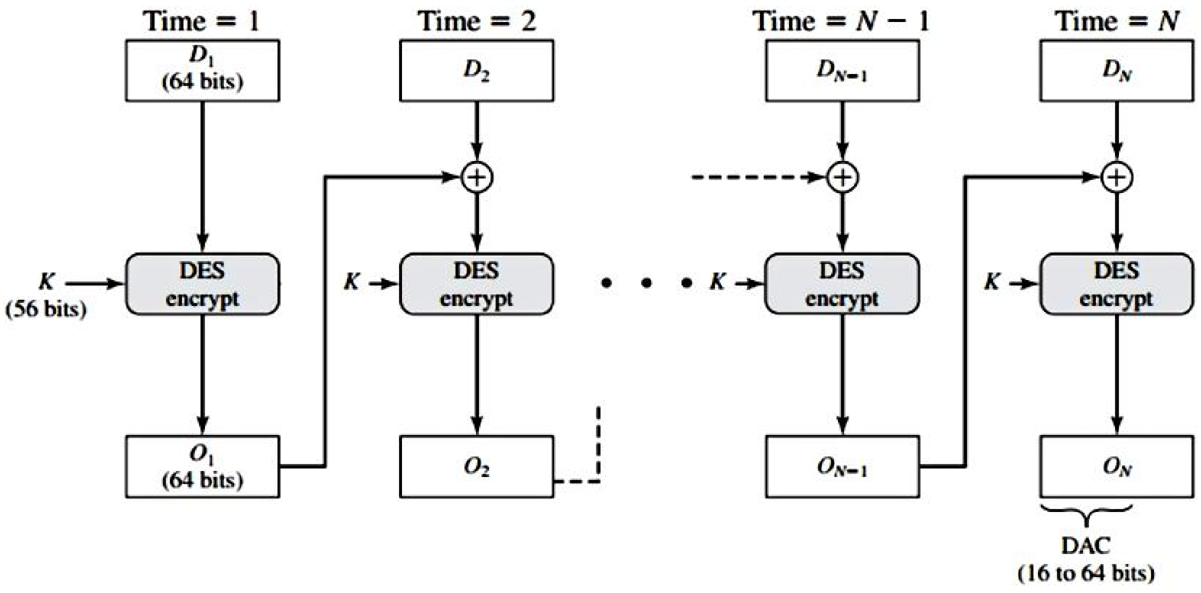


Figure : Data Authentication Algorithm (FIPS PUB 113)

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

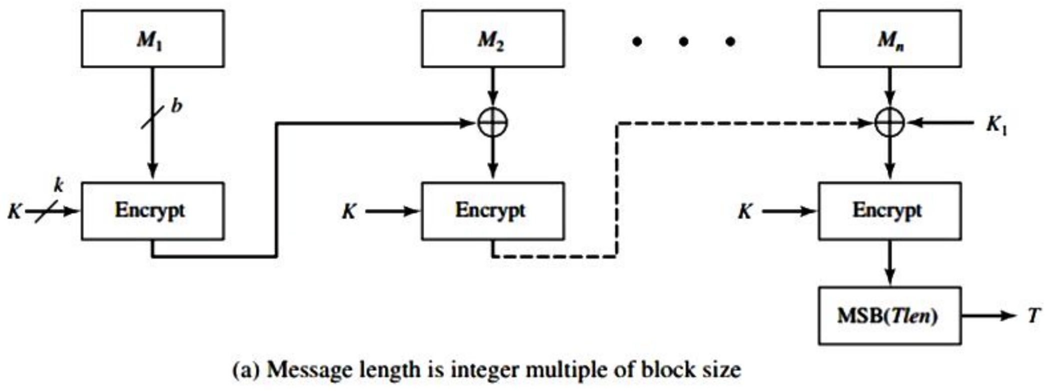
Q11. Explain briefly about CMAC.

Ans :

Cipher-Based Message Authentication Code (CMAC)

As was mentioned, DAA has been widely adopted in government and industry. Only messages of one fixed length of mn bits are processed, where n is the cipher block size and m is a fixed positive integer. As a simple example, notice that given the CBC MAC of a one-block message X, say T = MAC(K,X), the adversary immediately knows the CBC MAC for the two- block message X||(X{T) since this is once again T.

First, let us define the operation of CMAC when the message is an integer multiple n of the cipher block length b. For AES, b = 128, and for triple DES, b = 64. The message is divided into n blocks $(M_1, M_2, ..., M_n)$. The algorithm makes use of a k-bit encryption key K and an n-bit constant, K1. For AES, the key sizek is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits. CMAC is calculated as follows (Figure).



(a) Message length is integer multiple of block size

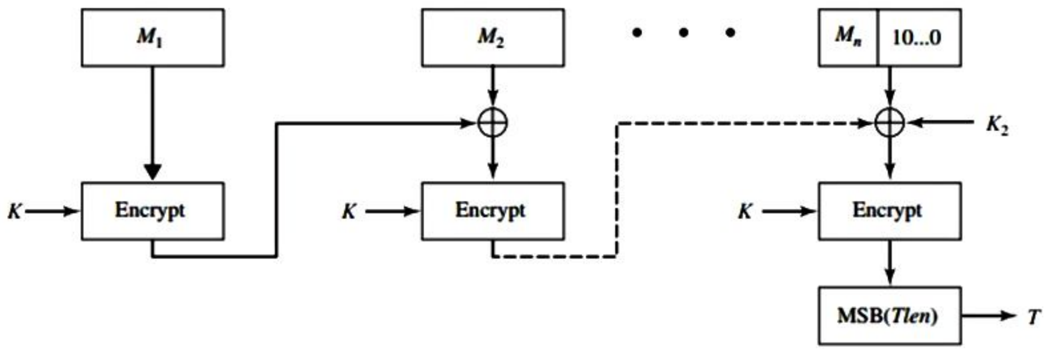


Figure : Cipher-Based Message Authentication Code (CMAC)

$C_1 = E(K, M_1)$

$C_2 = E(K, [M_2 \oplus C_1])$

$C_3 = E(K, [M_3 \oplus C_{2]}])$

$\vdots$

$C_n = E(K, [M_n \oplus C_{n-1} \oplus K_1])$

$T = MSB_{Tlen}(C_n)$

Where,

T = Message authetication code, also referred to as the tag

T_{len} = Bit length T

MSB,(X) = the S left most bits of the bit string X.

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

The two n-bit keys are derived from the k-bit encryption key as follows.

$L = E(K, 0^n)$

$K_1 = L \bullet x$

$K_2 = L \bullet x^2 = (L \bullet x) \bullet x$

where multiplication (•) is done in the finite field GF(2n) and x and x2 are first-and second-order polynomials that are elements of GF(2n). Thus, the binary representation of x consists of n-2 zeros followed by 10; the binary representation of x2 consists of n - 3 zeros followed by 100. The finite field is defined with respect to an irreducible polynomial that is lexicographically first among all such polynomials with the minimum possible number of nonzero terms. For the two approved block sizes, the polynomials are x64 + x4 + x3 + x + 1 and x128 + x7 + x2 + x + 1.

To generate K_1 and K_2, the block cipher is applied to the block that consists entirely of 0 bits. The first sub key is derived from the resulting ciphertext by a left shift of one bit and, conditionally, by XORing a constant that depends on the block size. The second subkey is derived in the same manner from the first subkey.

3.7.1 Digital Signatures

Q12. Explain briefly about digital signatures.

Ans :

Properties

Message authentication protects two parties who exchange messages from any third party. However, it doesnot protect the two parties against each other. Several forms of dispute between the two are possible.

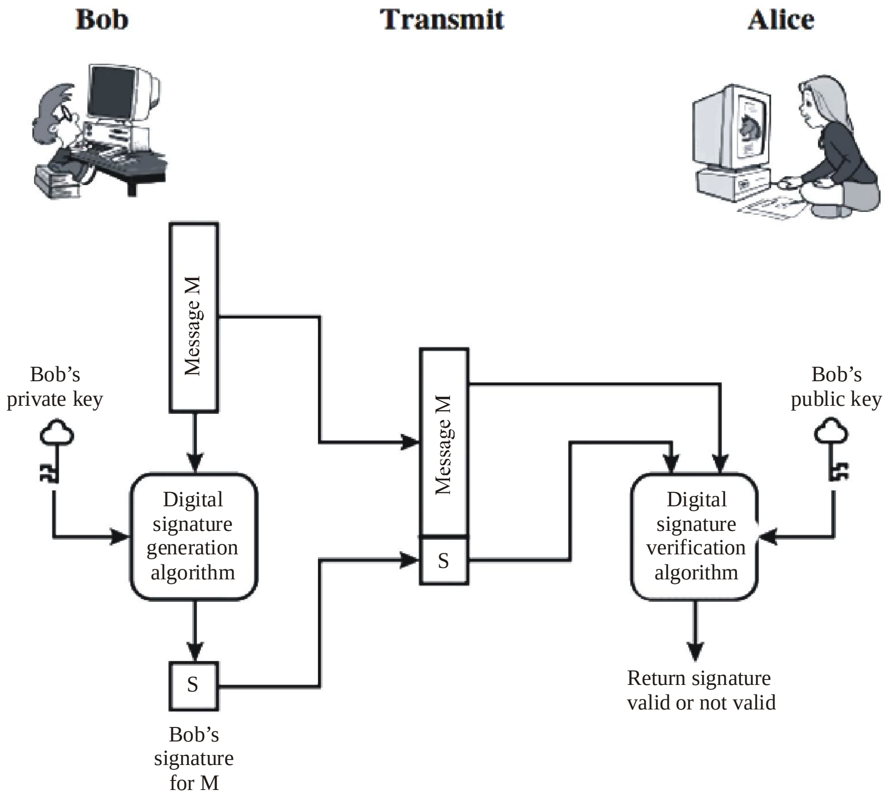

Bob
Transmit
Alice
Message M
Bob's private key
Digital signature generation algorithm
S
Bob's signature for M
Message M
S
Bob's public key
Digital signature verification algorithm
Return signature valid or not valid

Fig. : Generic Model of Digital Signature Process

For example, suppose that John sends an authenticated message to Mary, using one of the schemes of Figure. 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.

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

The digital signature must have the following properties:

➢ It must verify the author and the date and time of the signature.

➢ It must authenticate the contents at the time of the signature.

➢ It must be verifiable by third parties, to resolve disputes.

Thus, the digital signature function includes the authentication function.

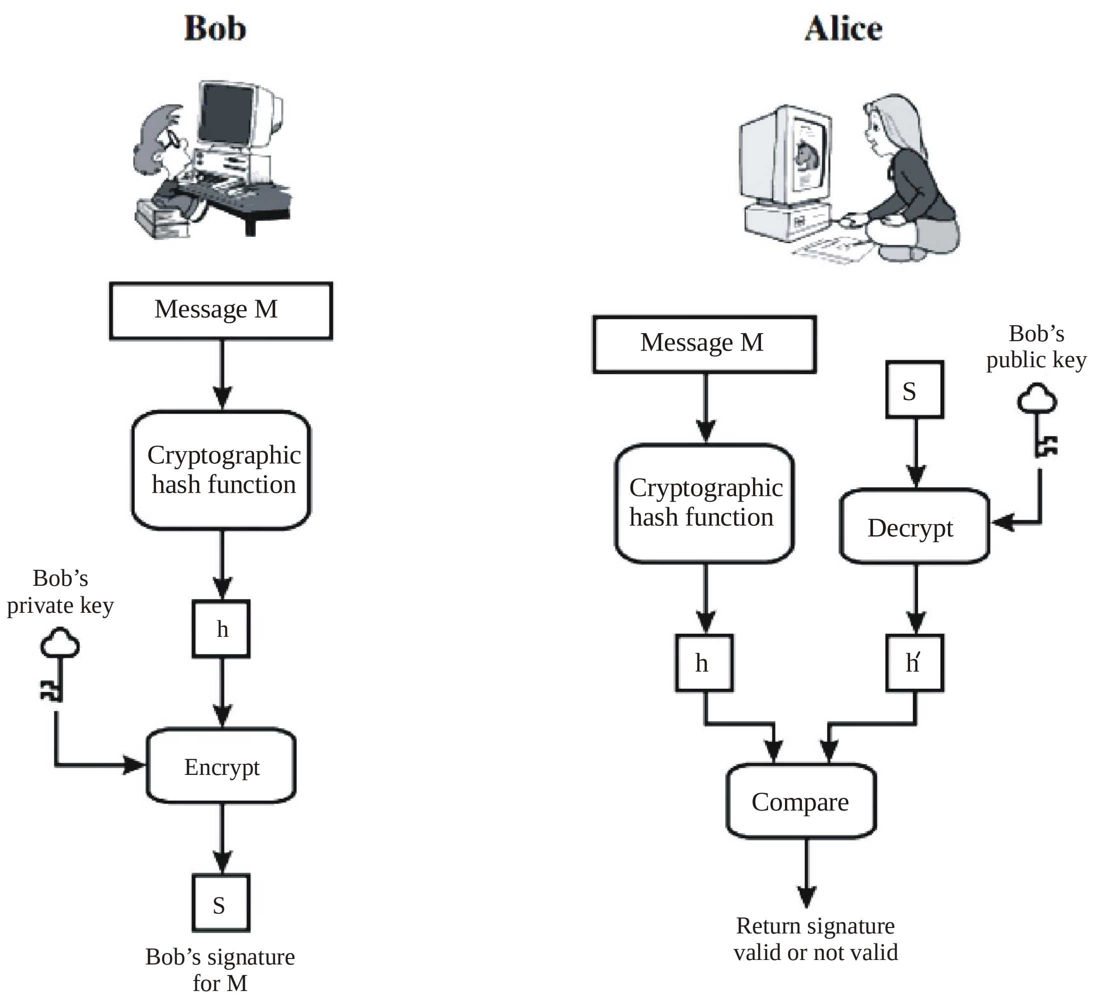


Fig. : Simplified Depiction of Essential Elements of Digital Signature Process

Attacks and Forgeries

The following types of attacks, in order of increasing severity. Here A denotes the user whose signature method is being attacked, and C denotes the attacker.

- Key-only attack: C only knows A's public key.
- Known message attack: C is given access to a set of messages and their signatures.
- Generic chosen message attack: C chooses a list of messages before attempting to breaks Signature scheme, independent of A's public key. C then obtains from A valid signatures for the chosen messages. The attack is generic, because it does not depend on A's public key; the same attack is used against everyone.
- Directed chosen message attack: Similar to the generic attack, except that the list of messages to besigned is chosen after C knows A's public key but before any signatures are seen.
- Adaptive chosen message attack: C is allowed to use A as an "oracle." This means the A may request signatures of messages that depend on previously obtained message–signature pairs.

Digital Signature Requirements

On the basis of the properties and attacks just discussed, 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.

3.7.2 Nist Digital Signatures Algorithm

Q13. Explain briefly about NIST digital structure algorithm.

Ans :

The Digital Signature Algorithm (DSA) is a variant of the ElGamal signature scheme

Key Generation

Key generation has two phases. The first phase is a choice of algorithm parameters which may be shared between different users of the system, while the second phase computes public and private keys for a single user.

Parameter Generation

Choose an approved cryptographic hash function H. In the original DSS, H was always SHA-1, but the stronger SHA-2 hash functions are approved for use in the current DSS.[5][9] The hash output may be truncated to the size of a key pair.

Decide on a key length L and N. This is the primary measure of the cryptographic strength of the key. The original DSS constrained L to be a multiple of 64 between 512 and 1,024 (inclusive). NIST 800-57 recommends lengths of 2,048 (or 3,072) for keys with security lifetimes extending beyond 2010 (or

2030), using correspondingly longer N.[10] FIPS 186-3 specifies L and N length pairs of (1,024, 160), (2,048, 224), (2,048, 256), and (3,072, 256).[4] N must be less than or equal to the output length of the hash H.

Choose an N-bit prime q.

Choose an L-bit prime p such that p " 1 is a multiple of q.

Choose g, a number whose multiplicative order modulo p is q. This may be done by setting g = h(p – 1)/q mod p for some arbitrary h (1 < h < p – 1), and trying again with a different h if the result comes out as 1. Most choices of h will lead to a usable g; commonly h = 2 is used.

The algorithm parameters (p, q, g) may be shared between different users of the system.

Per-user keys

Given a set of parameters, the second phase computes private and public keys for a single user:

Choose a secret key x by some random method, where 0 < x < q.

Calculate the public key y = gx mod p.

There exist efficient algorithms for computing the modular exponentiations h(p – 1)/q mod p and gx mod p, such as exponentiation by squaring.

Signing

Let H be the hashing function and m the message :

- Generate a random per-message value k where 1 < k < q
- Calculate $\tau = (g^k \bmod p) \bmod q$
- In the unlikely case that r = 0, start again with a different random k.
- Calculate $s = k^{-1} (H(m) + xr) \bmod q$
- In the unlikely case that s = 0, start again with a different random k.
- The signature is (r, s)

The first two steps amount to creating a new per-message key. The modular exponentiation here is the most computationally expensive part of the signing operation, and it may be computed before the message hash is known. The modular inverse k^ -1modq is the second most expensive part, and it may also be computed before the message hash is known. It may be computed using the extended Euclidean algorithm or using Fermat's little theorem as k^ q-2q.

Verifying

- Reject the signature if 0 < r < q or 0 < s < q is not satisfied.
- Calculate $w = s^{-1} \bmod q$
- Calculate $u_1 = H(m).w \bmod q$
- Calculate $u_2 = T.\ w \bmod q$
- Calculate $v = (g^{u1}\ y^{u2} \bmod p) \bmod q$
- The signature is invalid unless v = r

DSA is similar to the Elgamal signature scheme.

Correctness of the algorithm

The signature scheme is correct in the sense that the veritifer always accept signatures. This can be shown as follows.

First, if $g = h^{(p-1)/q} \bmod p$, it follows that $g^q = h^{p-1} \equiv 1 \bmod p$ by Femat's little theorem. Since g > 0 and q is prime g must have order q. The signer computes

$$s = k^{-1}(H(m) + xr) \bmod q$$

Thus

$$k \equiv H(m)s^{-1} + xrs^{-1}$$
$$\equiv H(m)w + xrw \pmod q$$

Since g has order q (mod p) we have

$$g^k \equiv g^{H(m)w}\ g^{xrw}$$
$$\equiv g^{H(m)w}\ g^{xrw}$$
$$\equiv g^{u1} y^{u2} \pmod p$$

Finally, the correctness of DSA follows from

$$r = (g^k \bmod p) \bmod q$$
$$= (g^{u1}, y^{u2} \bmod p) \bmod q$$
$$= v$$

UNIT IV

Transport-Level Security : Web Security Considerations, Secure Sockets Layer (SSL), Transport Layer Security (TLS), HTTPS,Secure Shell (SSH), E-Mail Security: Pretty Good Privacy, S/MIME. IP Security: IP Security Overview, IP Security Architecture, Encapsulating Security Payload, Combining Security Associations,Internet Key Exchange.Intruders, Virus and Firewalls: Intruders, Intrusion Detection, Password Management, Virus and Related Threats, Countermeasures, Firewall Design Principles, Types of Firewalls.

4.1 Transport - Level Security

4.1.1 Web Security Considerations

Q1. Write about the security considerations of internet.

Ans :

Web security Vulnerables

The Web is vulnerable to attacks on the Web servers over theInternet.

- The Web is increasingly serving as a highly visible outlet for corporate and product information and asthe platform for business transactions. Reputations can be damaged and money can be lost if the Web serversare subverted.
- Although Web browsers are very easy to use, Web servers are relatively easy to configure and manage,and Web content is increasingly easy to develop, the underlying software is extraordinarily complex.
- A Web server can be exploited as a launching pad into the corporation's or agency's entire computer complex. Once the Web server is subverted, an attacker may be able to gain access to data and systems not part of the Web itself but connected to the server at the local site.
- Casual and untrained (in security matters) users are common clients for Web-based services. Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge totake effective counter measures.

Web Security Threats

One way to group these threats is in terms of passive and active attacks. Passive attacks include eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to berestricted. Active attacks include impersonating another user, altering messages in transit between client andserver, and altering information on a Web site.

Another way to classify Web security threats is in terms of the location of the threat: Web server, Webbrowser, and network traffic between browser and server.

Web Traffic Security Approaches

A number of approaches to providing Web security are possible. The various approaches that have been considered are similar in the services they provide and, to some extent, in the mechanisms that they use, butthey differ with respect to their scope of applicability and their relative location within the TCP/IP protocolstack.

Figure 16.1 illustrates this difference. One way to provide Web security is to use IP security (IPsec). The advantage of using IPsec is that it is trans- parent to end users and applications and provides ageneral-purpose solution. Furthermore, IPsec includes a filtering capability so that only selected traffic need incur the overhead of IPsec processing.

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

4.1.2 Secure Socket Layer and Transport Layer Security

Q2. Explain about SSL architecture.

Ans :

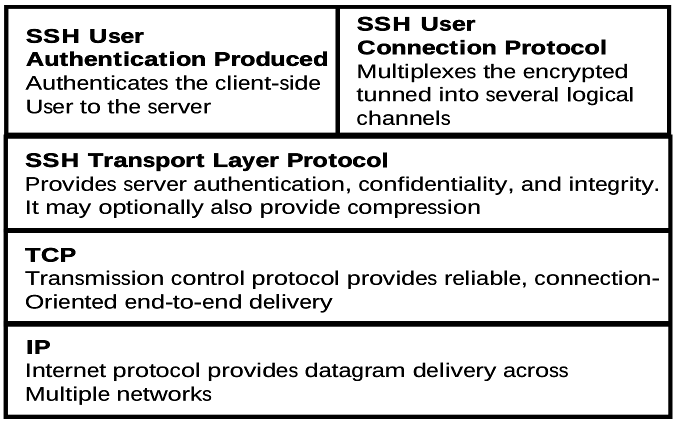
SSL Architecture

SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol but rather two layers of protocols, as illustrated in Figure 16.2.

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

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

- **Connection:** A connection is a transport (in the OSI layering model definition) that provides a suitable type of service. For SSL, such connections are peer-to-peer relationships. The connections are transient. Every connection is associated with one session.
- **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic security parameters which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection.


SSH User
Authentication Produced
Authenticates the client-side
User to the server
SSH User
Connection Protocol
Multiplexes the encrypted
tunned into several logical
channels
SSH Transport Layer Protocol
Provides server authentication, confidentiality, and integrity.
It may optionally also provide compression
TCP
Transmission control protocol provides reliable, connection-
Oriented end-to-end delivery
IP
Internet protocol provides datagram delivery across
Multiple networks

A session state is defined by the following parameters.

- **Session identifier:** An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
- **Peer certificate:** An X509.v3 certificate of the peer. This element of the state may be null.
- **Compression method:** The algorithm used to compress data prior to encryption.
- **Cipher spec:** Specifies the bulk data encryption algorithm (such as null, AES, etc.) and a hashalgorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such asthe hash_size.
- **Master secret :** 48-byte secret shared between the client and server.
- **Is resumable:** A flag indicating whether the session can be used to initiate newconnections.

  A connection state is defined by the following parameters.
- **Server and client random:** Byte sequences that are chosen by the server and client for each connection.
- **Server write MAC secret:** The secret key used in MAC operations on data sent by the server.
- **Client write MAC secret:** The secret key used in MAC operations on data sent by the client.
- **Server write key:** The secret encryption key for data encrypted by the server and decrypted by the client.

- **Client write key:** The symmetric encryption key for data encrypted by the client and decrypted by the server.
- **Initialization vectors:** When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter, the final cipher text block from each record is preserved for use as the IV with the following record.
- **Sequence numbers**: Each party maintains separate sequence numbers for transmitted and received messages for each connection.

Q3. Explain SSL Record Protocol.

Ans :

SSL Record Protocol

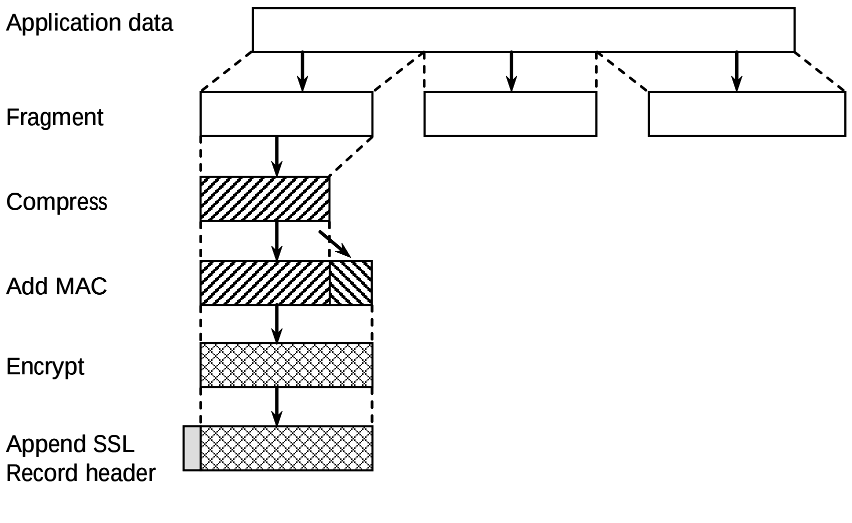
The SSL Record Protocol provides two services for SSL connections :

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

Figure 16.3 indicates the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users.

The first step is fragmentation. Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less. Next, compression is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes.1In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null.

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

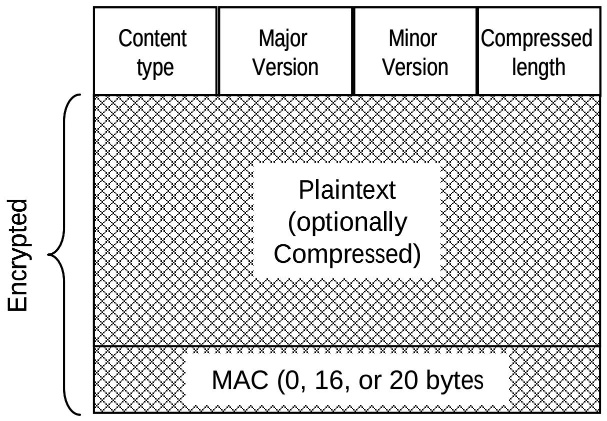


SSL Record Protocol Operation

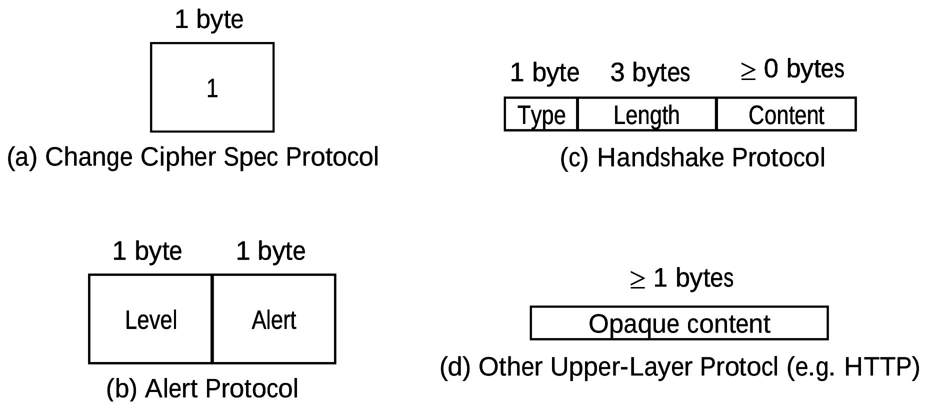
The final step of SSL Record Protocol processing is to prepare a header consisting of the following fields :

- Content Type (8 bits): The higher-layer protocol used to process the enclosed fragment.
- Major Version (8 bits): Indicates major version of SSL in use. For SSLv3, the value is 3.
- Minor Version (8 bits): Indicates minor version in use. For SSLv3, the value is 0.
- Compressed Length (16 bits): The length in bytes of the plaintext fragment (or compressed fragment if compression is used). The maximum value is 214 + 2048.

Figure 16.4 illustrates the SSL record format.



SSL Record Formula



SSL Record Protocol Payload

Q4. Discuss about Alter Protocol.

Ans :

Alert Protocol

The Alert Protocol is used to convey SSL-related alerts to the peer entity. As with other applications that use SSL, alert messages are compressed and encrypted, as specified by the current state.

Each message in this protocol consists of two bytes (Figure 16.5b). The first byte takes the value warning (1) or fatal (2) to convey the severity of the message. If the level is fatal, SSL immediately terminates the connection. Other connections on the same session may continue, but no new connections on this sessionmay be established. The second byte contains a code that indicates the specific alert. First, we list those alerts that are always fatal (definitions from the SSL specification):

- **unexpected_message:** An inappropriate message was received.
- **bad_record_mac:** An incorrect MAC was received.
- **decompression_failure:** The decompression function received improper input (e.g., unable to decompress or decompress to greater than maximum allowable length).
- **handshake_failure:** Sender was unable to negotiate an acceptable set of security parameters given theoptions available.
- **illegal_parameter:** A field in a handshake message was out of range or inconsistent with other fields.

The remaining alerts are the following :

- **close_notify:** Notifies the recipient that the sender will not send any more messages on this connection. Each party is required to send a close_notify alert before closing the write side of a connection.
- **no_certificate:** May be sent in response to a certificate request if no appropriate certificate isavailable.
- **bad_certificate:** A received certificate was corrupt (e.g., contained a signature that did not verify).
- **unsupported_certificate:** The type of the received certificate is not supported.
- **certificate_revoked:** A certificate has been revoked by its signer.
- **certificate_expired:** A certificate has expired.

e) **certificate_unknown :** Some other unspecified issue arose in processing the certificate, rendering itunacceptable.

Q5. Explain about Handshake protocol.

Ans :

Handshake Protocol

The most complex part of SSL is the Handshake Protocol. This protocol allows the server and client toauthenticate each other and to negotiate an encryption and MAC algorithm and cryptographic keys to be used to protect data sent in an SSL record. The Handshake Protocol is used before any application data is transmitted.

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

a) **Type (1 byte):** Indicates one of 10 messages. Table 16.2 lists the defined message types.

b) **Length (3 bytes):** The length of the message in bytes.

c) **Content (Ú 0 bytes):** The parameters associated with this message; these arelisted in Table 16.2.

Figure 16.6 shows the initial exchange needed to establish a logical connection between client and server. Theexchange can be viewed as having four phases.

Phase 1. Establish Security Capabilities

This phase is used to initiate a logical connection and to establish the security capabilities that will be associated with it. The exchange is initiated by the client, which sends aclient_hello message with the following parameters:

d) **Version:** The highest SSL version understood by the client.

e) **Random:** A client-generated random structure consisting of a 32-bit timestamp and 28 bytes generated by a secure random number generator. These values serve as nonces and are used during keyexchange to prevent replay attacks.

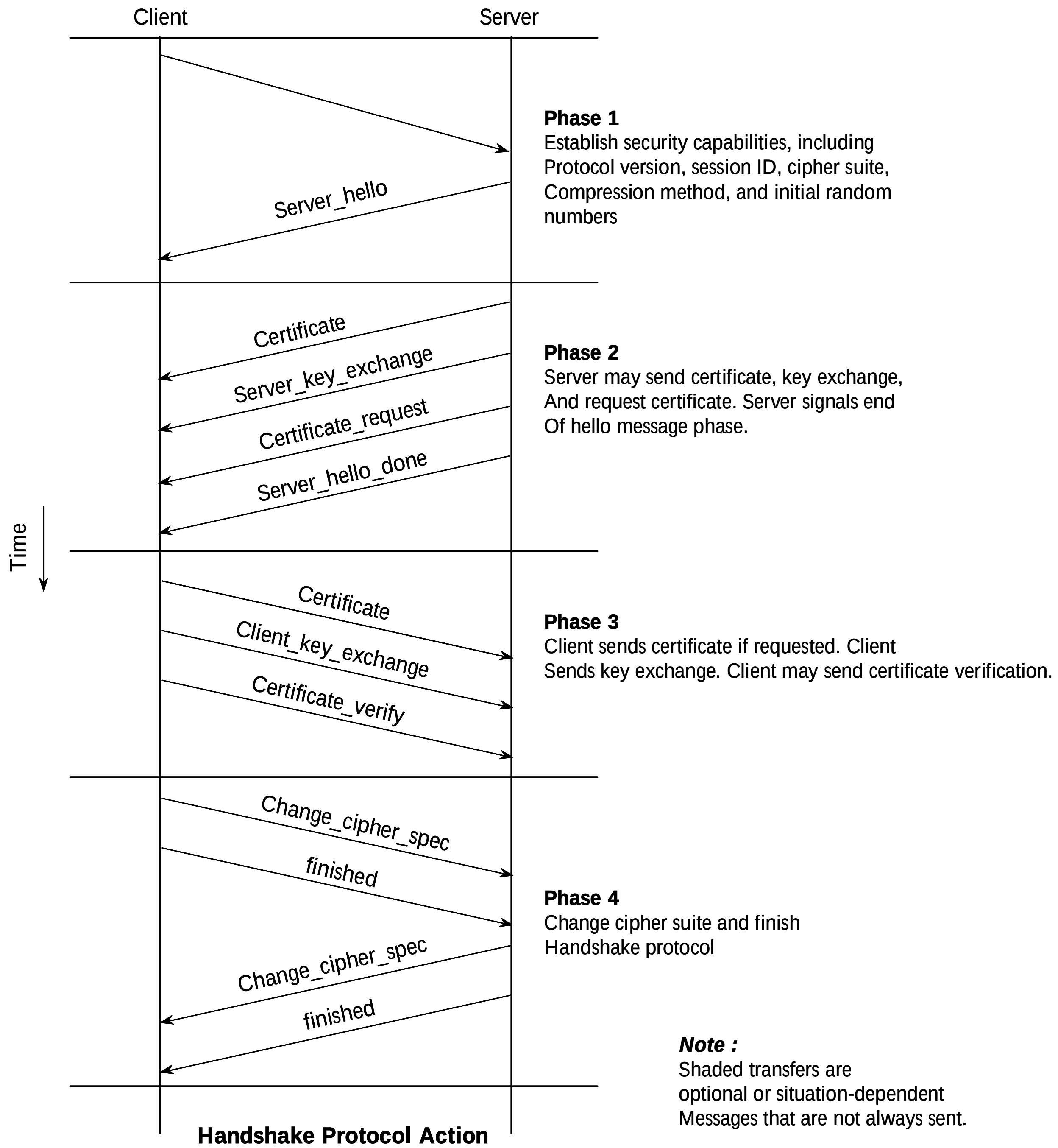
Message	Parameters
hello_request	Null
client_hello	verson, random, session id, cipher suite, compression method
server_hello	verson, random, session id, cipher suite, compression method
certificate	chain of X.509v3 certificate
server_key_exchange	parameters, signature
certificate_request	type.authotities
server_done	Null
certificate_verify	signature
client_key_exchange	Parameters, signature
finished	hash value

Table : SSL Handshake Protocol Message Types

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

After sending the client_hello message, the client waits for the server_hello message, which contains the same parameters as the client_hello message. For the server_hello message, the following conventions apply. The Version field contains the lower of the versions suggested by the client and the highest supported by the server.

The Random field is generated by the server and is independent of the client's Random field. If the SessionID field of the client was nonzero, the same value is used by the server; otherwise the server's SessionID field contains the value for a new session. The CipherSuite field contains the single cipher suite selected by the server from those proposed by the client. The Compression field contains the compression method selected by the server from those proposed by the client.


Client
Server
Server_hello
Phase 1
Establish security capabilities, including
Protocol version, session ID, cipher suite,
Compression method, and initial random
numbers
Certificate
Server_key_exchange
Certificate_request
Server_hello_done
Phase 2
Server may send certificate, key exchange,
And request certificate. Server signals end
Of hello message phase.
Time
Certificate
Client_key_exchange
Certificate_verify
Phase 3
Client sends certificate if requested. Client
Sends key exchange. Client may send certificate verification.
Change_cipher_spec
finished
Change_cipher_spec
finished
Phase 4
Change cipher suite and finish
Handshake protocol
Note :
Shaded transfers are
optional or situation-dependent
Messages that are not always sent.

Handshake Protocol Action

Phase 2. Server Authentication and Key Exchange

The server begins this phase by sending its certificate if it needs to be authenticated; the message contains one or a chain of X.509 certificates. The certificate message is required for any agreed-on key exchange method except anonymous Diffie-Hellman. Note that if fixed Diffie- Hellman is used, this certificate message functions as the server's key exchange message because it contains the server's public Diffie-Hellman parameters.

Next, a server_key_exchange message may be sent if it is required. It is not required in two instances: (1)The server has sent a certificate with fixed Diffie-Hellman parameters or (2) a RSA key exchange is to be used. The server_key_exchange message is needed for the following:

- **Anonymous Diffie-Hellman:** The message content consists of the two global Diffie-Hellman values (a prime number and a primitive root of that number) plus the server's public Diffie-Hellman key .
- **Ephemeral Diffie-Hellman:** The message content includes the three Diffie- Hellman parameters provided for anonymous Diffie-Hellman plus a signature of those parameters.
- **RSA key exchange :**Accordingly, the client cannot simply send a secret key encrypted with the server's public key. Instead, the server must create a temporary RSA public/private key pair and use the server_key_exchange message tosend the public key.

Phase 3. Client Authentication and Key Exchange

Upon receipt of the server_done message, the client should verify that the server provided a valid certificate (if required) and check that the server_hello parameters are acceptable. If all is satisfactory, the client sends one or more messagesback to the server.

If the server has requested a certificate, the client begins this phase by sending a certificate message. If no suitable certificate is available, the client sends a no_certificate alert instead.

Next is the client_key_exchange message, which must be sent in this phase.

Phase 4. Finish

This phase completes the setting up of a secure connection. The client sends achange_cipher_spec message and copies the pending CipherSpec into the current CipherSpec.. The finished message verifies that the key exchange and authentication processes were successful.The content of the finished message is the concatenation of two hash values :

MD5(master_secret | pad2 | MD5(handshake_messages | Sender | master_secret | pad1))

SHA(master_secret | pad2 | SHA(handshake_messages | Sender | master_secret | pad1))

4.2 HTTPS

Q6. Explain briefly about HHTPS protocol.

Ans :

HTTPS (HTTP over SSL) refers to the combination of HTTP and SSL to implement secure communication between a Web browser and a Web server. The HTTPS capability is built into all modern Web browsers. Its use depends on the Web server supporting HTTPS communication. For example, search engines do not support HTTPS.

When HTTPS is used, the following elements of the communication are encrypted :

- URL of the requested document
- Contents of the document
- Contents of browser forms (filled in by browser user)

- Cookies sent from browser to server and from server to browser
- Contents of HTTP header

Connection Initiation

For HTTPS, the agent acting as the HTTP client also acts as the TLS client. The client initiates a connection to the server on the appropriate port and then sends the TLS Client Hello to begin the TLS handshake. When the TLS handshake has finished, the client may then initiate the first HTTP request. All HTTP data is to be sent as TLS application data. Normal HTTP behavior, including retained connections, should be followed.

We need to be clear that there are three levels of awareness of a connection in HTTPS. At the HTTP level, an HTTP client requests a connection to an HTTP server by sending a connection request to the next lowest layer.

Connection Closure

An HTTP client or server can indicate the closing of a connection by including the following line in an HTTP record: Connection: close. This indicates that the connection will be closed after this record is delivered.

The closure of an HTTPS connection requires that TLS close the connection with the peer TLS entity on there mote side, which will involve closing the underlying TCP connection. At the TLS level, the proper way to close a connection is for each side to use the TLS alert protocol to send a close_notify alert. TLS implementations must initiate an exchange of closure alerts before closing a connection.

HTTP clients also must be able to cope with a situation in which the underlying TCP connection is terminated without a prior close_notify alert and without a Connection: close indicator. Such a situation could be due to a programming error on the server or a communication error that causes the TCP connection to drop. However, the unannounced TCP closure could be evidence of some sort of attack. So the HTTPS client should issue some sort of security warning when this occurs.

4.3 Secure Shell (SSH)

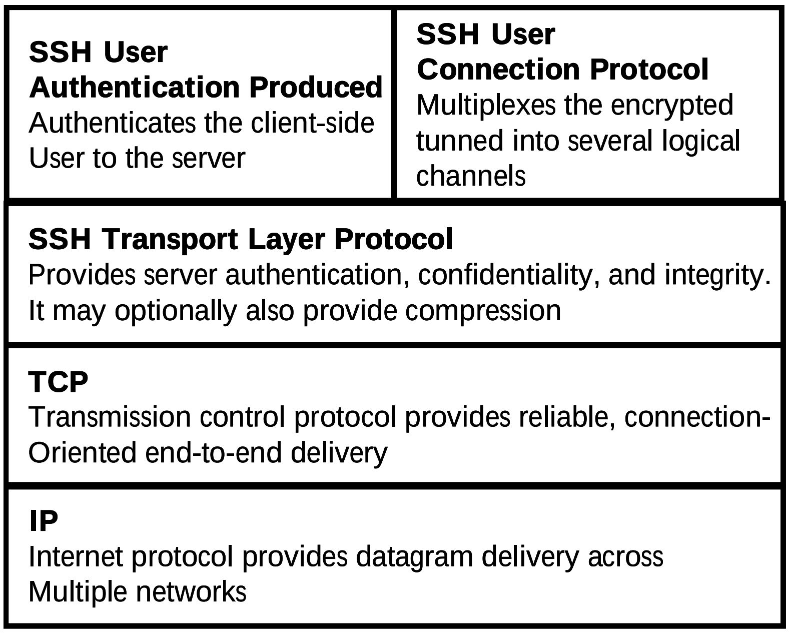
Q7. Write about SSH.

Ans :

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

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

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



SSH User Authentication Produced
Authenticates the client-side User to the server

SSH User Connection Protocol
Multiplexes the encrypted tunned into several logical channels

SSH Transport Layer Protocol
Provides server authentication, confidentiality, and integrity. It may optionally also provide compression

TCP
Transmission control protocol provides reliable, connection-Oriented end-to-end delivery

IP
Internet protocol provides datagram delivery across Multiple networks

- **User Authentication Protocol :** Authenticates the user to the server.
- **Connection Protocol :** Multiplexes multiple logical communications channels over a single, underlying SSH connection.

Transport Layer Protocol

HOST KEYS Server authentication occurs at the transport layer, based on the server possessing a public/private key pair. A server may have multiple host keys using multiple different asymmetric encryption algorithms. Multiple hosts may share the same host key. RFC 4251 dictates two alternative trust models that can be used:

1. The client has a local database that associates each host name (as typed by the user) with the corresponding public host key. This method requires no centrally administered infrastructure and no third-party coordination.
2. The host name-to-key association is certified by a trusted certification authority (CA). The client only knows the CA root key and can verify the validity of all host keys certified by accepted CAs.

PACKET EXCHANGE Figure 16.9 illustrates the sequence of events in the SSH Transport Layer Protocol. First, the client establishes a TCP connection to the server. This is done via the TCP protocol and is not part of the Transport Layer Protocol. Once the connection is established, the client and server exchange data, referred to as packets, in the data field of a TCP segment. Each packet is in the following format (Figure 16.10).

- **Packet length:** Length of the packet in bytes, not including the packet length and MAC fields.
- **Padding length:** Length of the random padding field.
- **Payload:** Useful contents of the packet. Prior to algorithm negotiation, this field is uncompressed. If compression is negotiated, then in subsequent packets, this field is compressed.

- **Random padding:** Once an encryption algorithm has been negotiated, this field is added. It contains random bytes of padding so that that total length of the packet (excluding the MAC field) is a multiple of the cipher block size, or 8 bytes for a stream cipher.
- **Message authentication code (MAC):** If message authentication has been negotiated, this field contains the MAC value. The MAC value is computed over the entire packet plus a sequence number,excluding the MAC field. The sequence number is an implicit 32-bit packet sequence that is initialized to zero for the first packet and incremented for every packet. The sequence num- ber is not included in the packetsent over the TCP connection.

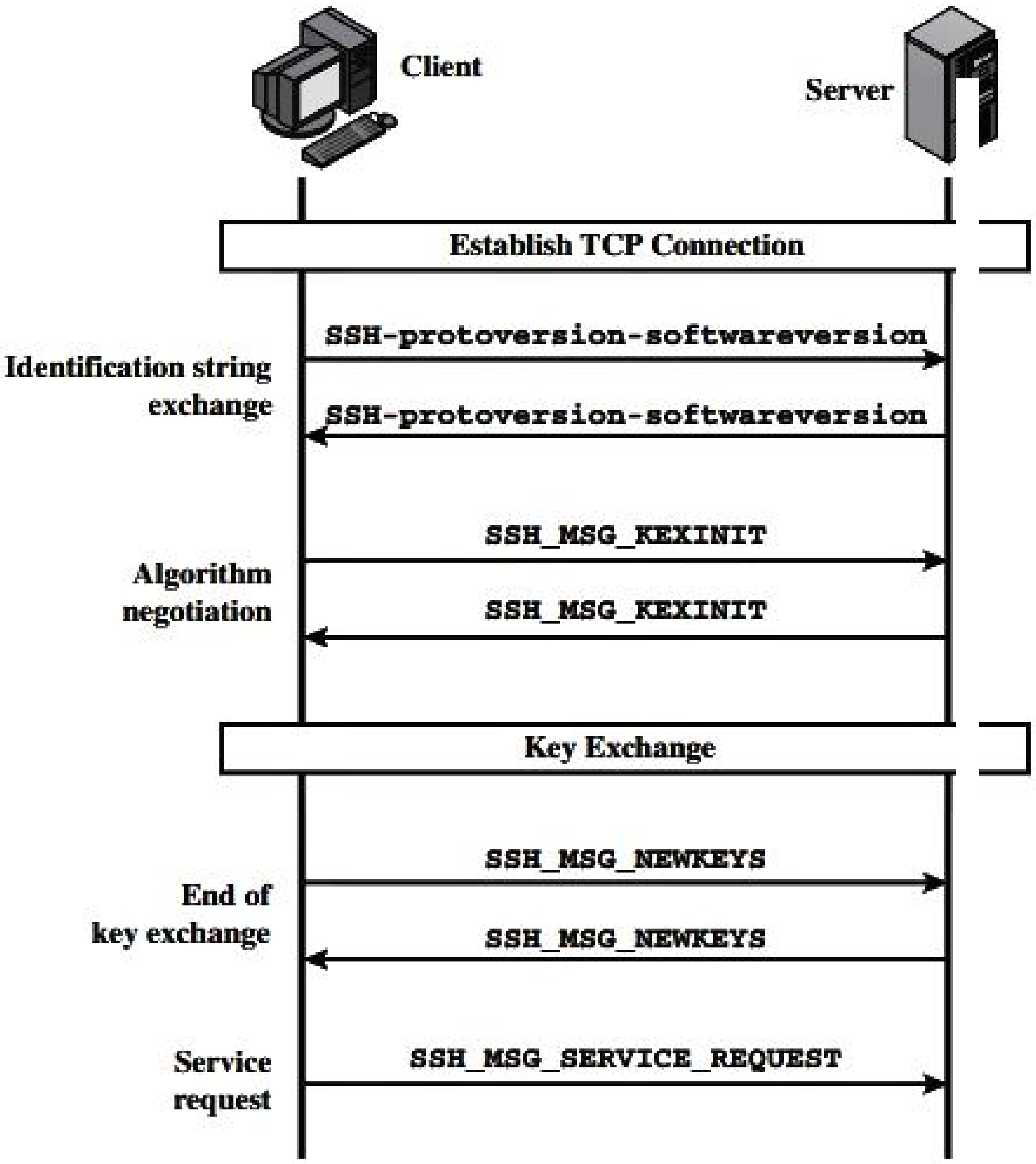
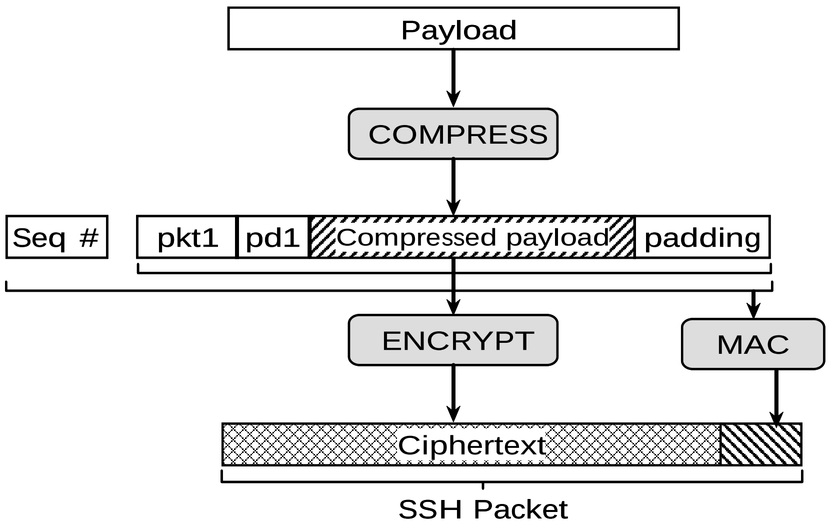


Figure 16.9 SSH Transport Layer Protocol Packet Exchanges



pktl = packet length
pdl = paddling length

SSH Transport Layer Protocol packet Formation

User Authentication Protocol

The User Authentication Protocol provides the means by which the client is authen- ticated to the server.

Message Types and Formats

Three types of messages are always used in the User Authentication Protocol. Authentication requests from the client have the format :

byte-	SSH_MSG_USERAUTH_REQUEST (50)
string	user name string service name string method name
...	method specific fields

where user name is the authorization identity the client is claiming, service name is the facility to which the client is requesting access and method name is the authentication method being used in this request. The first byte has decimal value 50, which isinterpreted as SSH_MSG_USERAUTH_REQUEST.

If the server either (1) rejects the authentication request or (2) accepts therequest but requires one or more additional authentication methods, the server sends a message with the format:

byte	SSH_MSG_USERAUTH_FAILURE (51)
name-list	authentications that can continue
boolean	partial success

Message Exchange

The message exchange involves the following steps.

1. The client sends a SSH_MSG_USERAUTH_REQUEST with a requested method of none.

2. The server checks to determine if the user name is valid. If not, the server returns SSH_ MSG_USERAUTH_FAILURE with the partial success value of false. If the user name is valid, the serverproceeds to step 3.
3. The server returns SSH_MSG_USERAUTH_FAILURE with a list of one or more authentication methods to be used.
4. The client selects one of the acceptable authentication methods and sends a SSH_MSG_USE RAUTH_REQUEST with that method name and the required method-specific fields.
5. If the authentication succeeds and more authentication methods are required, the server proceeds tostep 3, using a partial success value of true. If the authentication fails, the server proceeds to step 3, using a partialsuccess value of false.
6. When all required authentication methods succeed, the server sends a SSH _ MSG _ USERAUTH _ SUCCESS message, and the Authentication Protocol is over.

Authentication Methods

The server may require one or more of the following authentication methods.

- **publickey :** The details of this method depend on the public-key algorithm chosen. In essence, the client sends a message to the server that contains the client's public key, with the message signed by theclient's private key. When the server receives this message, it checks whether the supplied key is acceptablefor authentication and, if so, it checks whether the signature is correct.
- **password :** The client sends a message containing a plaintext password, which is protected byencryption by the Transport Layer Protocol.
- **hostbased :** Authentication is performed on the client's host rather than the client itself.

Connection Protocol

The SSH Connection Protocol runs on top of the SSH Transport Layer Protocol and assumes that a secureauthentication connection is in use.

Channel Mechanism

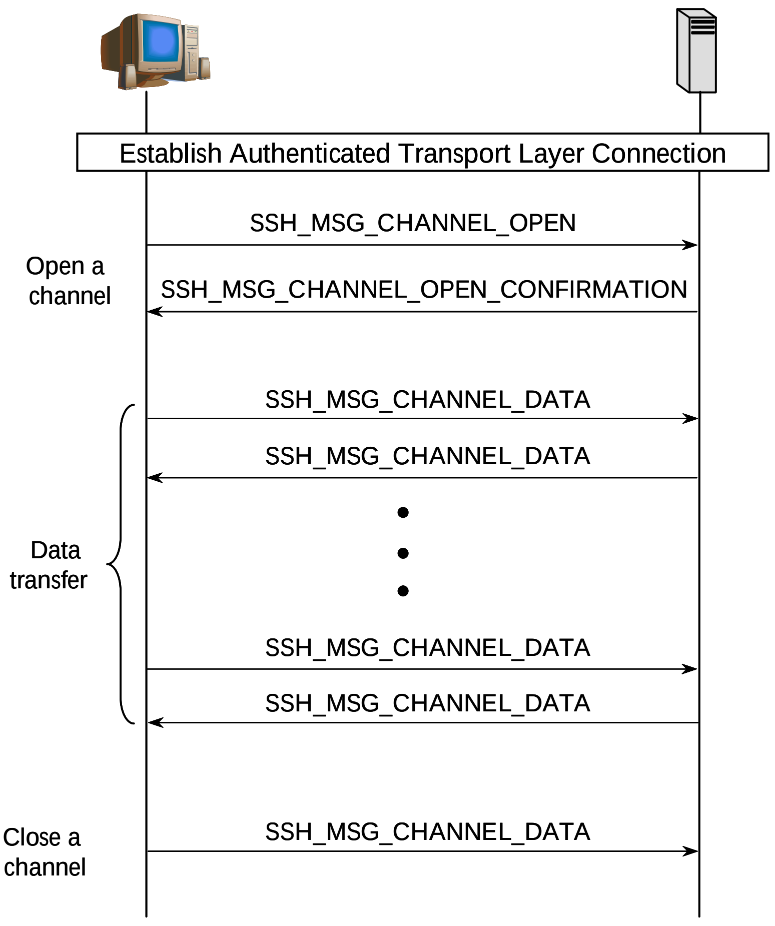
All types of communication using SSH, such as a terminal session, are supportedusing separate channels. Either side may open a channel. For each channel, each side associates a uniquechannel number, which need not be the same on both ends. Channels are flow controlled using a window mechanism. No data may be sent to a channel until a message is received to indicate that window space isavailable.

Figure 16.11 provides an example of Connection Protocol Message Exchange.

Channel Types

Four channel types are recognized in the SSH Connection Protocol specification.

Session : The remote execution of a program. The program may be a shell, an application such as file transferor e-mail, a system command, or some built-in subsystem. Once a session channel is opened, subsequentrequests are used to start the remote program.



Example SSH Connection Protocol Message Exchange

- **x11 :** This refers to the X Window System, a computer software system and net- work protocol that provides a graphical user interface (GUI) for networked computers. X allows applications to run on a networkserver but to be displayed on a desktop machine.
- **forwarded-tcpip :** This is remote port forwarding, as explained in the next sub- section.
- **direct-tcpip :** This is local port forwarding, as explained in the next subsection.

- **x11 :** This refers to the X Window System, a computer software system and net- work protocol that provides a graphical user interface (GUI) for networked computers. X allows applications to run on a networkserver but to be displayed on a desktop machine.
- **forwarded-tcpip :** This is remote port forwarding, as explained in the next sub- section.
- **direct-tcpip :** This is local port forwarding, as explained in the next subsection.

Port Forwardning

One of the most useful features of SSH is port forwarding. In essence, port forwarding provides the ability to convert any insecure TCP connection into a secure SSH connection. This is also referredto as SSH tunneling. We need to know what a port is in this context. A port is an identifier of a user of TCP.So, any application that runs on top of TCP has a port number. Incoming TCP traffic is delivered to theappropriate application on the basis of the port number.An application may employ multiple port numbers.

Figure 16.12 illustrates the basic concept behind port forwarding. We have a client application that isidentified by port number x and a server application identi- fied by port number y. At some point, the client application invokes the local TCP entity and requests a connection to the remote server on port y. The localTCP entity negotiates a TCP connection with the remote TCP entity, such that the connection links local portx to remote port y.

To secure this connection, SSH is configured so that the SSH Transport Layer Protocol establishes a TCP connection between the SSH client and server entities with TCP port numbers a and b, respectively. A secure SSH tunnel is established over this TCP connection. Traffic from the client at port x is redirected tothe localSSH entity and travels through the tunnel where the remote SSH entity delivers the data to the server application on port y. Traffic in the other direction is similarly redirected.

SSH supports two types of port forwarding: local forwarding and remote for- warding. Local forwardingallows the client to set up a “hijacker” process. This will intercept selected application-level traffic and redirect it from an unsecured TCP connection to a secure SSH tunnel. SSH is configured to listen on selected ports. SSH grabs all traffic using a selected port and sends it through an SSH tunnel. On the otherend, the SSH server sends the incoming traffic to the destination port dic- tated by the client application.

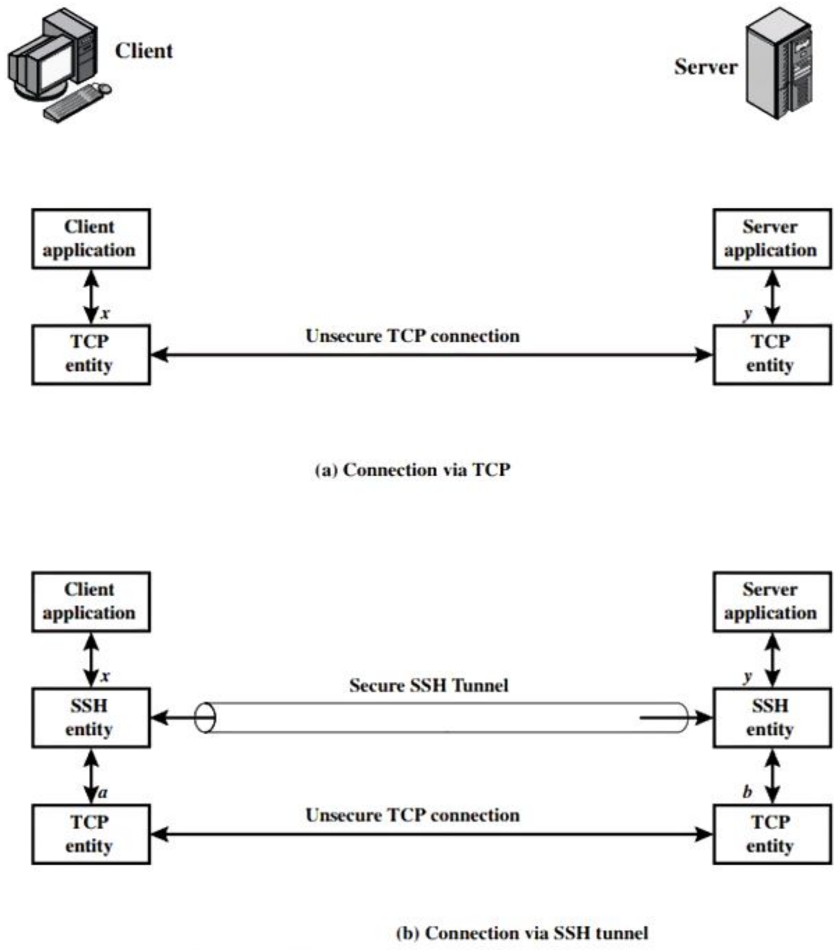


Figure 16.12 SSH Transport Layer Packet Exchanges

We can secure this traffic in the following way :

1. The SSH client sets up a connection to the remote server.
2. Select can unused local port number, say 9999, and configure SSH to accept traffic from thisport destined for port 110 on the server.
3. The SSH client informs the SSH server to create a connection to the destina tion, in this case mail server port 110.
4. The client takes any bits sent to local port 9999 and sends them to the server inside theencrypted SSH session. The SSH server decrypts the incoming bits and sends the plaintext to port 110.
5. In the other direction, the SSH server takes any bits received on port 110 and sends theminside the SSH session back to the client, who decrypts and sends them to the process connected to port 9999.

4.4 E-Mail Security

4.4.1 Pretty Good Privacy

Q8. What is PGP (Pretty Good Privacy) discuss it ?

Ans :

PGP has grown explosively and is now widely used. A number of reasons can be cited for this growth.

1. It is available free worldwide in versions that run on a variety of platforms, including Windows,UNIX, Macintosh, and many more.
2. It is based on algorithms that have survived extensive public review and are con- sidered extremely secure. Specifically, the package includes RSA, DSS, and Diffie-Hellman for public-key encryption.
3. It has a wide range of applicability, from corporations that wish to select and enforce a standardizedscheme for encrypting files and messages to individuals who wish to communicate securely with othersworldwide over the Internet and other networks.
4. It was not developed by, nor is it controlled by, any governmental or standards organization.
5. PGP is now on an Internet standards track

Notation

Most of the notation used in this chapter has been used before, but a few terms are new. It is perhaps best to summarize those at the beginning. The following symbols are used.

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

The actual operation of PGP, as opposed to the management of keys, consists of four services :

Authentication

Figure 18.1a illustrates the digital signature service provided by PGP. This is the digital signature scheme discussed in Chapter 13 and illustrated in Figure 13.2. The sequence is as follows.

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

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

Table 18.1 : Summary of PGP Services

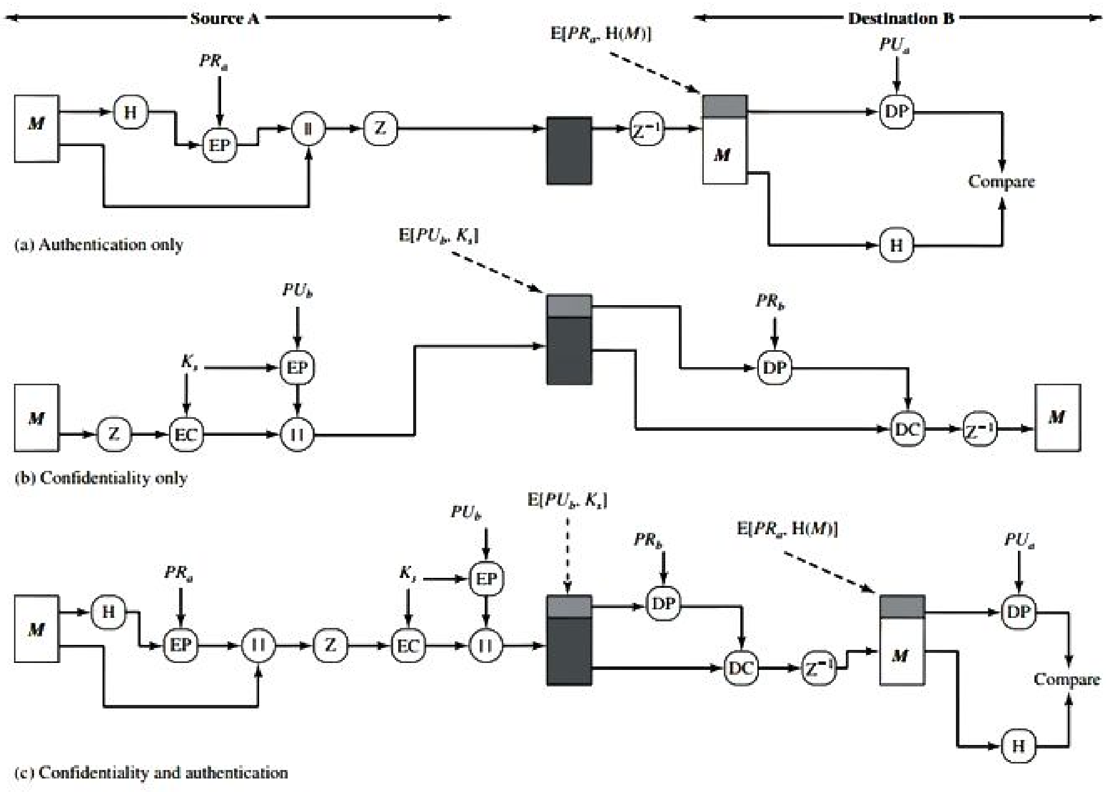


Figure 18.1 PGP Cryptographic Functions

Confidentiality

Another basic service provided by PGP is confidentiality, which is provided by encryptingmessages to be transmitted or to be stored locally as files. In both cases, the symmetric encryption algorithmCAST-128 may be used. Alternatively, IDEA or 3DES may be used. The 64-bit cipher feedback (CFB) mode is used.

Figure 18.1b illustrates the sequence, which can be described as follows.

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

As an alternative to the use of RSA for key encryption, PGP provides an option referred to as Diffie-Hellman.

Compression

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

1. The signature is generated before compression for two reasons:
   a. It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification.

2. Message encryption is applied after compression to strengthen cryptographic security. Becausethe compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

E-Mail Compatability

When PGP is used, at least part of the block to be transmitted is encrypted. If only thes signature service is used, then the message digest is encrypted . If theconfidentiality service is used, the message plus signature are encrypted..

The scheme used for this purpose is radix-64 conversion. Each group of three octets of binary data ismapped into four ASCII characters.

Cryptographic Keys and Key Rings

PGP makes use of four types of keys: one-time session symmetric keys, public keys, private keys, andpassphrase-based symmetric keys :

1. A means of generating unpredictable session keys is needed.
2. We would like to allow a user to have multiple public-key/private-key pairs. One reason is that theuser may wish to change his or her key pair from time to time. When this happens, any messages in the pipeline will be constructed with an obsolete key.
3. Each PGP entity must maintain a file of its own public/private key pairs as well as a file ofpublic keys of correspondents.

Session Key Generation

Each session key is associated with a single message and is used only for the purpose of encrypting and decrypting that message. Recall that message encryption/decryption is done witha symmetric encryption algorithm. CAST-128 and IDEA use 128-bit keys; 3DES uses a 168-bit key. For the following discussion, we assume CAST-128.

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 cipherfeedback mode, the CAST-

Key Indentifiers

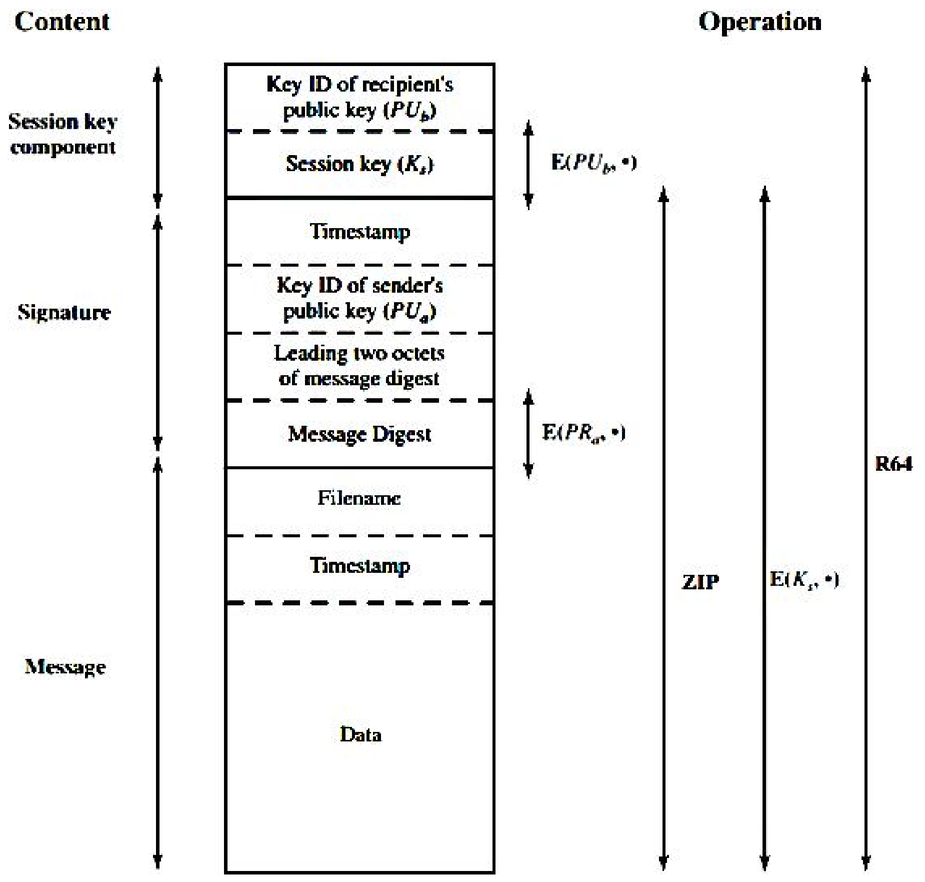
The solution adopted by PGP is to assign a key ID to each public key that is, with very high probability, unique within a user ID.

A message consists of three components: the message component, a signature (optional), and a session key component (optional).

The **message component** includes the actual data to be stored or transmitted, as well as a filename and atimestamp that specifies the time of creation.

The **signature component** includes the following.

- **Timestamp :** The time at which the signature was made.
- **Message digest :** The 160-bit SHA-1 digest encrypted with the sender's private signature key. The digest is calculated over the signature timestamp concatenated with the data portion of the message component.



Notation :

E(PU_b, •)	=	encryption with user b’s public key
E(PR_a, •)	=	encryption with user a’s private key
E(K_s, •)	=	encryption with session key
ZIP	=	Zip compression function
R64	=	Radix-64 conversion function

Fig. : General Format PGP Message (from A to B)

Public-Key Management

PGP provides a structure for solving this problem with several suggested options that may be used. Because PGP is intended for use in a variety of formal and informal environments.

Approaches to Public-key Management, for example, if A got the key from a bulletin board system (BBS) that was used by B to post the public key but that has been compromised by C. The result is that two threats now exist. First, C can send messages to Aand forge B’s signature so that A will accept the message as coming from B. Second, any encrypted messagefrom A to B can be read by C.

A number of approaches are possible for minimizing the risk that a user’s public-key ring contains falsepublic keys. Suppose that A wishes to obtain a reliable public key for B. The following are some approachesthat could be used.

1. Physically get the key from B. B could store her public key (PUb) on a floppy disk and hand it to A. A could then load the key into his system from the floppy disk. This is a very secure method but has obvious practical limitations.

2. Verify a key by telephone. If A can recognize B on the phone, A could call B and ask her to dictate thekey, in radix-64 format, over the phone. As a more practical alternative, B could transmit her key in an e-mailmessage to A. A could have PGP generate a 160-bit SHA-1 digest of the key and display it in hexadecimal format;.
3. Obtain B's public key from a mutual trusted individual D. For this purpose, the introducer, D, creates asigned certificate. The certificate includes B's public key, the time of creation of the key, and a validity period forthe key. D generates an SHA-1 digest of this certificate, encrypts it with her private key, and attaches the signature to the certificate. Because only D could have created the signature, no one else can create a false public key and pretend that it is signed by D. The signed certificate could be sent directly to A by B or D, or itcould be posted on a bulletin board.
4. Obtain B's public key from a trusted certifying authority. Again, a public-key certificate is createdand signed by the authority. A could then access the authority, providing a user name and receiving a signedcertificate.

4.5 S/MIME

Q9. Write about S/MIME.

Ans :

Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security.

RFC 5322

RFC 5322 defines a format for text messages that are sent using electronic mail. It has been the standard forInternet-based text mail messages and remains in common use. In the RFC 5322 context, messages are viewed ashaving an envelope and contents.The envelope contains whatever information is needed to accomplishtransmission and delivery. The contents compose the object to be delivered to the recipient.

Multipurpose Internet Mail Extensions

Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP), defined in RFC 821.

1. SMTP cannot transmit executable files or other binary objects. A number of schemes are in usefor converting binary files into a text form that can be used by SMTP mail systems, including the popular UNIX UUencode/UUdecode scheme.
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 consistentset of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle nontextual data included in X.400 messages.

Overview

The MIME specification includes the following elements.

1. Five new message header fields are defined, which may be included in an RFC 5322 header. Thesefields 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 for- mat into a form that isprotected from alteration by the mail system.

The five header fields defined in MIME are

- **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 that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropri- ate manner.
- **Content-Transfer-Encoding:** Indicates the type of transformation that has been used to represent thebody 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 isnot readable (e.g., audio data).

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 capabilitycan 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.

4.6 IP Security

4.6.1 IP Security Overview

10. Explain IP security applications.

Ans :

Applications of IPsec

IPsec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include :

- **Secure branch office connectivity over the Internet:** A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet andreduce its need for private networks, saving costs and network management overhead.

- **Secure remote access over the Internet:** An end user whose system is equipped with IP securityprotocols can make a local call to an Internet Service Provider (ISP) and gain secure access to a companynetwork. This reduces the cost of toll charges for traveling employees and telecommuters.
- **Establishing extranet and intranet connectivity with partners:** IPsec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.
- **Enhancing electronic commerce security:** Even though some Web and electronic commerce applicationshave built-in security protocols, the use of IPsec enhances that security. IPsec guarantees that all traffic designatedby the network administrator is both encrypted and authenticated, adding an additional layer of security towhatever is provided at the application layer.

Figure 19.1 is a typical scenario of IPsec usage.

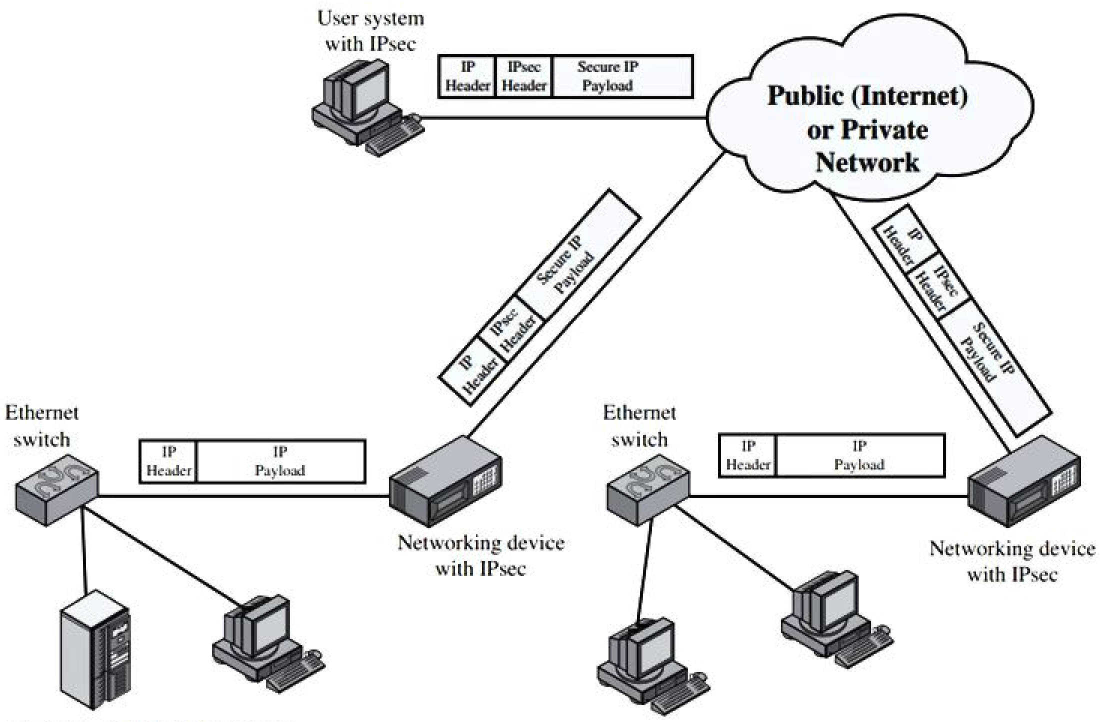


Figure 19.1 An IP Security Scenario

Q11. What are the benefits of IPSec ?

Ans :

Benefits of IPsec

Some of the benefits of IPsec :

- When IPsec is implemented in a firewall or router, it provides strong security that can be applied to alltraffic crossing the perimeter.
- IPsec in a firewall is resistant to bypass if all traffic from the outside must use IP and the firewall is the only means of entrance from the Internet into the organization.

- IPsec is below the transport layer (TCP, UDP) and so is transparent to applications. There is noneed to change software on a user or server system when IPsec is implemented in the firewall or router.
- IPsec can be transparent to end users. There is no need to train users on security mechanisms, issue keyingmaterial on a per-user basis, or revoke keying material when users leave the organization.
- IPsec can provide security for individual users if needed. This is useful for offsite workers and for settingup a secure virtual subnetwork within an organization for sensitive applications.

Q12. Explain IP security services.

Ans :

IPsec Services

IPsec provides security services at the IP layer by enabling a system to select required security protocols,determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required toprovide the requested services. lists the following services:

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets (a form of partial sequence integrity)
- Confidentiality (encryption)
- Limited traffic flow confidentiality

Q13. Write about transport mode and tunnel mode.

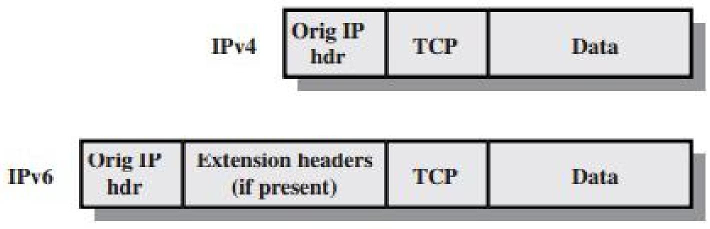
Ans :

Transport and Tunnel Modes

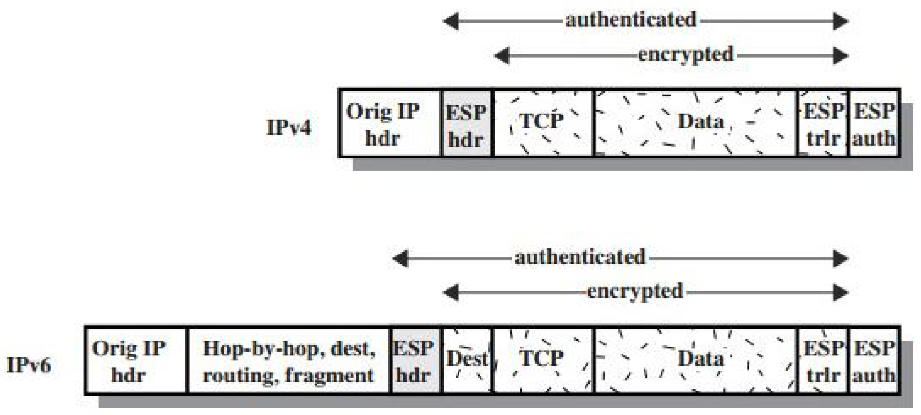
Transport Mode

Transport mode provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet.. When ahost runs AH or ESP over IPv4, the payload is the data that normally follow the IP header. For IPv6, thepayload is the data that normally follow both the IP header and any IPv6 extensions headers that arepresent, with the possible exception of the destination options header, which may be included in theprotection.

ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header. AH in transport mode authenticates the IP payload and selected portions of the IP header.



(a) Before Applying ESP

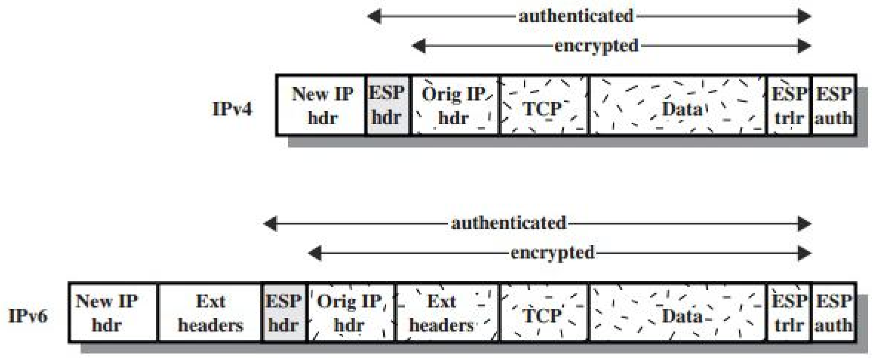


(b) Transport Mode

Tunnel Mode

Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH orESP 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, 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 a securityassociation (SA) are a security gateway, such as a firewall or router that implements IPsec. With tunnel mode, a number of hosts on networks behind firewalls may engage in secure communications without implementing IPsec. The unprotected packets generated by such hosts are tunneled through external networks by tunnel mode SAs set up by the IPsec software in the firewall or secure router at the boundary of the local network.



(c) Tunnel Mode

Figure 19.8 Scope of ESP Encryption and Authentication

4.6.2 IP Security Architecture

Q14. Describe IP Security Architecture.

Ans :

- **Architecture:** Covers the general concepts, security requirements, definitions, and mechanisms defining IPSec technology.
- **Encapsulating Security Payload (ESP):** Covers the packet format and general issues related to the use of the ESP for packet encryption and, optionally, authentication.
- **Authentication Header (AH):** Covers the packet format and general issues related to the use of AH for packet authentication.
- **Encryption Algorithm:** A set of documents that describe how various encryption algorithms are used for ESP.

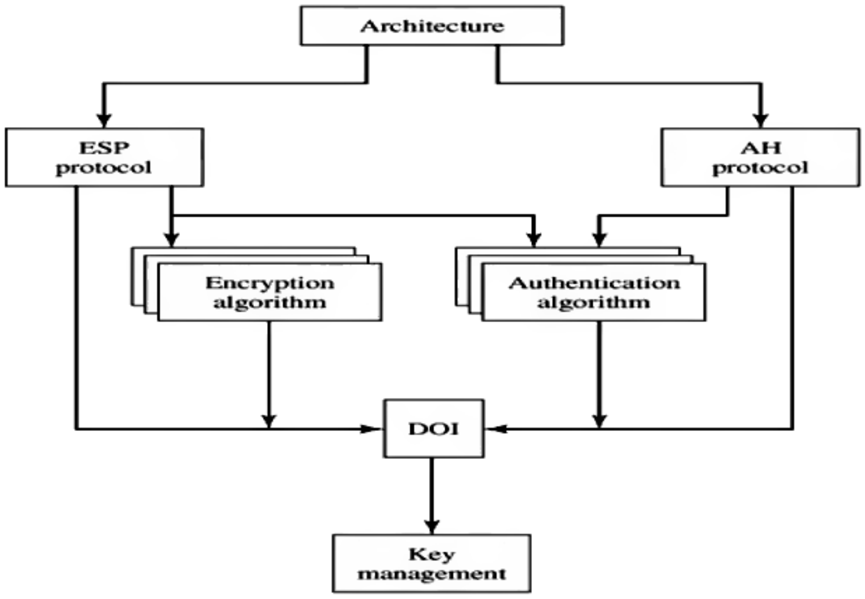


Fig. : 1.2 IPSec Document Overview

- **Authentication Algorithm:** A set of documents that describe how various authentication algorithms are used for AH and for the authentication option of ESP.
- **Key Management:** Documents that describe key management schemes.

Domain of Interpretation (DOI): Contains values needed for the other documents to relate to each other. These include identifiers for approved encryption and authentication algorithms, as well as operational parameters such as key lifetime.

4.6.3 Encapsulating Security Payload

Q15. Explain about ESP (Encapsulaitng Security Payload).

Ans :

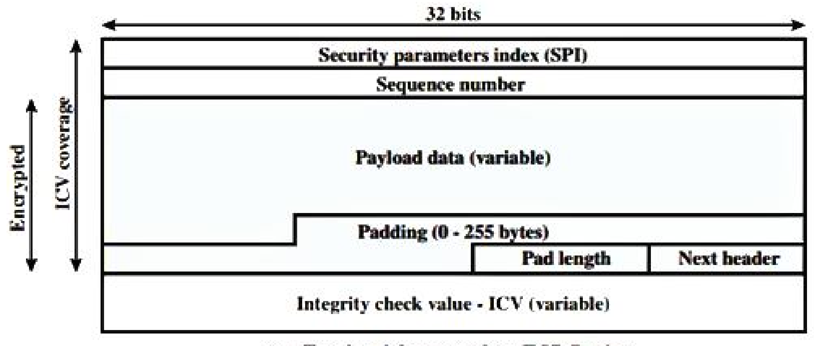
ESP can be used to provide confidentiality, data origin authentication, connection- less integrity, an anti-replay service (a form of partial sequence integrity), and (lim- ited) traffic flow confidentiality. The set of services provided depends on options selected at the time of Security Association (SA) establishment and on the location of the implementation in a network topology.

ESP can work with a variety of encryption and authentication algorithms, including authenticated encryption algorithms such as GCM.

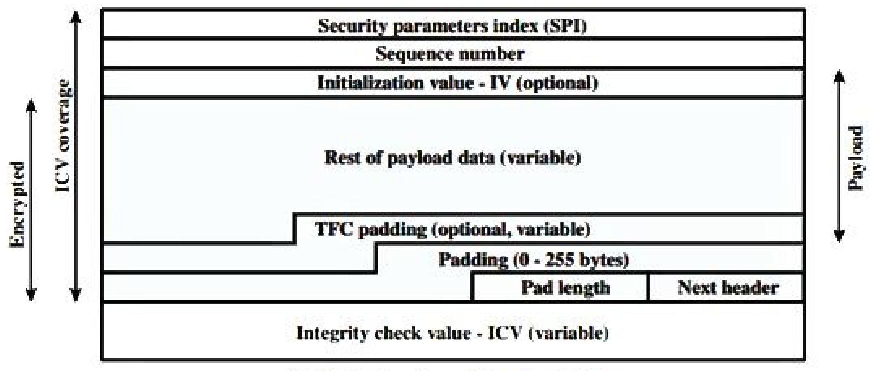
ESP Format

Figure 19.5 a shows the top-level format of an ESP packet. It contains the following fields.

- **Security Parameters Index (32 bits):** Identifies a security association.
- **Sequence Number (32 bits):** A monotonically increasing counter value; this provides an anti-replayfunction, as discussed for AH.
- **Payload Data (variable):** This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
- **Padding (0 – 255 bytes):** The purpose of this field is discussed later.
- **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field.
- **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifyingthe first header in that payload (for example, an extension header in IPv6, or an upper-layer protocol such asTCP).



(a) Top-level format of an ESP Packet


Security parameters index (SPI)
Sequence number
Initialization value - IV (optional)
Rest of payload data (variable)
TFC padding (optional, variable)
Padding (0 - 255 bytes)
Pad length
Next header
Integrity check value - ICV (variable)
Encrypted
ICV coverage
Payload

(b) Substructure of payload data

Figure 19.5 ESP Packet Format

- **Integrity Check Value (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field.

Encryption and Authentication Algorithms

The Payload Data, Padding, Pad Length, and Next Header fields are encrypted by the ESP service. If the algorithm used to encrypt the payload requires crypto- graphic synchronization data, such as an initialization vector (IV), then these data may be carried explicitly at the beginning of the Payload Data field. If included, an IV is usually not encrypted, although it is often referred to as being part of the ciphertext.

Padding

The Padding field serves several purposes:

- If an encryption algorithm requires the plaintext to be a multiple of some number of bytes (e.g., the multiple of a single block for a block cipher), the Padding field is used to expand the plaintext (consisting of the Payload Data, Padding, Pad Length, and Next Header fields) to the required length.
- The ESP format requires that the Pad Length and Next Header fields be right aligned within a 32-bit word. Equivalently, the ciphertext must be an integer multiple of 32 bits. The Padding field is used to assure this alignment.

Anti-Replay Service

A replay attack is one in which an attacker obtains a copy of an authenticated packet and later transmits it to the intended destination. The receipt of duplicate, authenticated IP packets may disrupt service in some way or may have some other undesired consequence. The Sequence Number field is designed to thwart such attacks. First, we discuss sequence number generation by the sender, and then we look at how it is processed by the recipient.

4.6.5 Combining Security Associations

16. Explain how to combine security associations.

Ans :

Security associations may be combined into bundles in two ways :

- **Transport Adjacency :** Refers to applying more than one security protocol to the same IP packet without invoking tunneling. This approach to combining AH and ESP allows for only one level of combination; further nesting yields no added benefit since the processing is performed at one IPsec instance:the (ultimate) destination.
- **Iterated tunneling:** Refers to the application of multiple layers of security protocols effected through IP tunneling. This approach allows for multiple levels of nesting, since each tunnel can originate or terminate at a different IPsec site along the path.

The two approaches can be combined, for example, by having a transport SA between hosts travel part of the way through a tunnel SA between security gateways.

Authentication Plus Confidentiality

Encryption and authentication can be combined in order to transmit an IP packet that has both confidentiality and authentication between hosts. We look at several approaches.

ESP with Authentication Option

This approach is illustrated in Figure 19.8. In this approach, the user first applies ESP to the data to be protected and then appends the authentication data field. There are actually two subcases :

- **Transport mode ESP :** Authentication and encryption apply to the IP payload delivered to the host, but the IP header is not protected.
- **Tunnel mode ESP:** Authentication applies to the entire IP packet delivered to the outer IPdestination address (e.g., a firewall), and authentication is performed at that destination. The entire inner IP packet is protected by the privacy mechanism for delivery to the inner IP destination.

For both cases, authentication applies to the ciphertext rather than the plaintext.

Transport Adjacency

Another way to apply authentication after encryption is to use two bundled transport SAs, with the inner being an ESP SA and the outer being an AH SA. In this case, ESP is used without itsauthentication option.

Transport-Tunnel Bundle

The use of authentication prior to encryption might be preferable for several reasons. First, because the authentication data are protected by encryption, it is impossible for anyone to intercept the message and alter the authentication data without detection. Second, it may be desirable to storethe authentication information with the message at the destination for later reference.

Basic Combinations of Security Associations

The IPsec Architecture document lists four examples of combinations of SAs that must be supported by compliant IPsec hosts (e.g., workstation, server) or security gateways (e.g. firewall, router). These areillustrated in Figure 19.10.

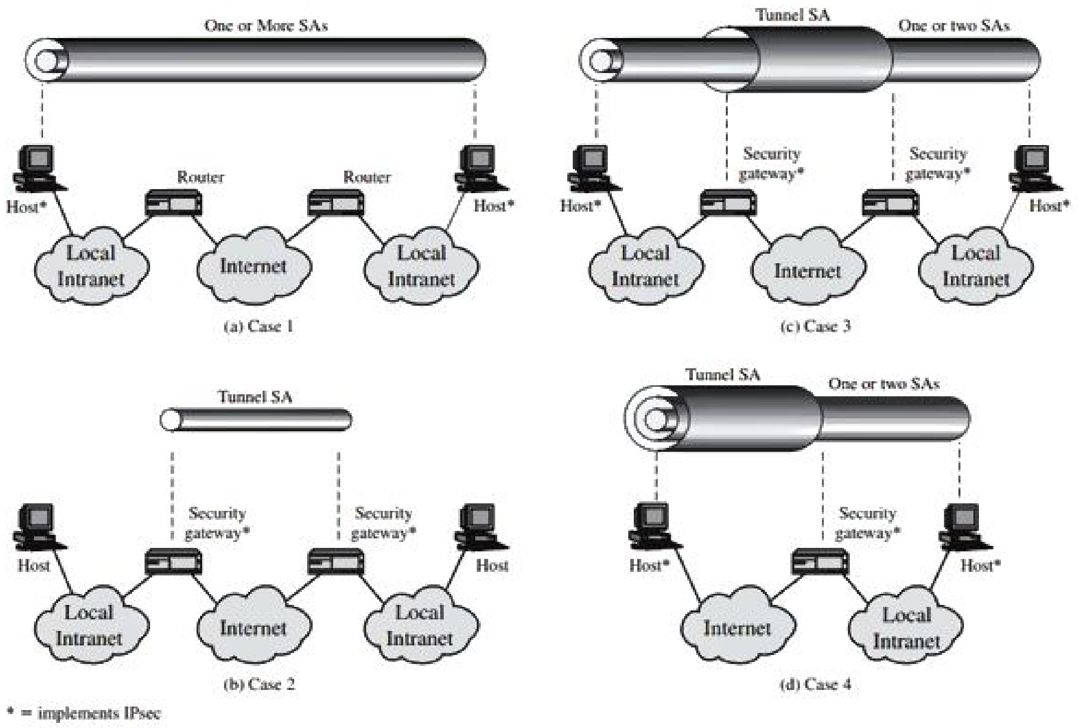


Figure 19.10 Basic Combinations of Security Associations

The lower part of each case in the figure represents the physical connectivity of the elements; the upper part representslogical connectivity via one or more nested SAs. Each SA can be either AH or ESP. For host-to-host SAs, the mode may be either transport or tunnel; otherwise it must be tunnel mode.

Case 1. All security is provided between end systems that implement IPsec. For any two end systems tocommunicate via an SA, they must share the appropriate secret keys. Among the possible combinations are

- AH in transport mode
- ESP in transport mode
- ESP followed by AH in transport mode (an ESP SA inside an AH SA)
- Any one of a, b, or c inside an AH or ESP in tunnel mode

Case 2. Security is provided only between gateways (routers, firewalls, etc.) and no hosts implement IPsec. This case illustrates simple virtual private network support.

Case 3. This builds on case 2 by adding end-to-end security. The same combinations discussed for cases 1 and 2 are allowed here. The gateway-to-gateway tunnel provides either authentication, confidentiality, or both for all traffic between end systems.

Case 4. This provides support for a remote host that uses the Internet to reach an organization's firewall and thento gain access to some server or workstation behind the firewall.

4.6.6 Internet Key Exchange

Q17. Write about Inter Key Exchange.

Ans :

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 integrity and confidentiality. The IPsec Architecture document mandates support for two types of keymanagement:

- **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 aframework for Internet key management and provides the specific protocol support, including formats, fornegotiation of security attributes.

Key Determination Protocol

IKE key determination is a refinement of the Diffie-Hellman key exchange algo- rithm. Recall that Diffie-Hellman involves the following interaction between users A and B. There is prior agreement on two globalparameters: q, a large prime number; and a, a primitive root of q. A selects a random integer XA as its privatekey and transmits to B its public key VA = aXA mod q. Similarly, B selects a random integer XB as its privatekey and transmits to A its public key VB = aXB mod q. Each side can now compute the secret session key:

K = (VB)XA mod q = (VA)XB mod q = aXAXB mod q

The Diffie-Hellman algorithm has two attractive features :

- Secret keys are created only when needed. There is no need to store secret keys for a long period oftime, exposing them to increased vulnerability.
- The exchange requires no pre-existing infrastructure other than an agreement on the global parameters.

IKE key determination is designed to retain the advantages of Diffie- Hellman, while countering its weaknesses.

Features of IKE Determination

The IKE key determination algorithm is characterized by five important features :

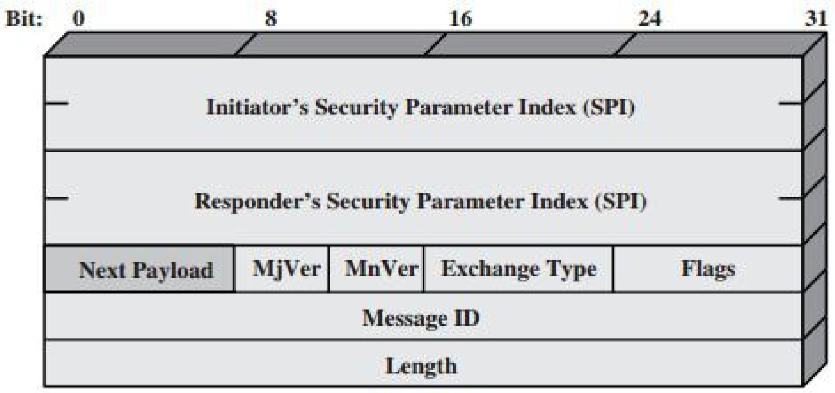
1. It employs a mechanism known as cookies to thwart clogging attacks.
2. It enables the two parties to negotiate a group; this, in essence, specifies the global parameters of theDiffie-Hellman key exchange.
3. It uses nonces to ensure against replay attacks.
4. It enables the exchange of Diffie-Hellman public key values.
5. It authenticates the Diffie-Hellman exchange to thwart man-in-the-middle attacks.

Header and Payload Formats

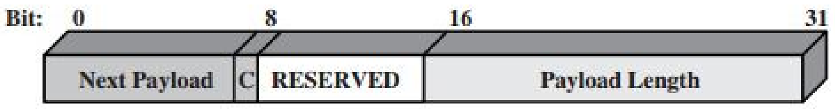
IKE defines procedures and packet formats to establish, negotiate, modify, and delete security associations.**IKE HEADER FORMAT** An IKE message consists of an IKE header followed by one or more payloads. All of thisis carried in a transport protocol. The specification dictates that implementations must support the use of UDPfor the transport protocol.

Figure 19.12a shows the header format for an IKE message. It consists of the following fields.

- **Initiator SPI (64 bits):** A value chosen by the initiator to identify a unique IKE security association(SA).
- **Responder SPI (64 bits):** A value chosen by the responder to identify a unique IKE SA.
- **Next Payload (8 bits):** Indicates the type of the first payload in the message; payloads are discussed inthe next subsection.
- **Major Version (4 bits):** Indicates major version of IKE in use.
- **Minor Version (4 bits):** Indicates minor version in use.


Bit: 0
8
16
24
31
Initiator's Security Parameter Index (SPI)
Responder's Security Parameter Index (SPI)
Next Payload
MjVer
MnVer
Exchange Type
Flags
Message ID
Length

(a) IKE header



(b) Generic Payload header

Figure 19.12 IKE Formats

- **Exchange Type (8 bits):** Indicates the type of exchange; these are discussed later in this section.
- **Flags (8 bits):** Indicates specific options set for this IKE exchange. Three bits are defined so far. Theinitiator bit indicates whether this packet is sent by the SA initiator. The version bit indicates whether the transmitter is capable of using a higher major version number than the one currently indicated. The response bit indicates whether this is a response to a message containing the same message ID.
- **Message ID (32 bits):** Used to control retransmission of lost packets and matching of requestsand responses.
- **Length (32 bits):** Length of total message (header plus all payloads) in octets.

IKE Payload Types

All IKE payloads begin with the same generic payload header shown in Figure 19.12b. The Next Payload field has a value of 0 if this is the last payload in the message; otherwise its value is thetype of the next payload. The Payload Length field indicates the length in octets of this payload, including the generic payload header.

These elements are formatted as substructures within the payload as follows.

- **Proposal:** This substructure includes a proposal number, a protocol ID (AH, ESP, or IKE), anindicator of the number of transforms, and then a transform substructure.
- **Transform:** Different protocols support different transform types. The trans- forms are used primarilyto define cryptographic algorithms to be used with a particular protocol.
- **Attribute:** Each transform may include attributes that modify or complete the specification of the transform. An example is key length.

  1. The Key Exchange payload can be used for a variety of key exchange tech- niques, including Oakley,Diffie-Hellman, and the RSA-based key exchange used by PGP. The Key Exchange data field contains the data required to generate a session key and is dependent on the key exchange algorithm used.
  2. The Identification payload is used to determine the identity of communicating peers and may be used fordetermining authenticity of information. Typically the ID Data field will contain an IPv4 or IPv6 address.
  3. The Certificate payload transfers a public-key certificate. The Certificate Encoding field indicates the type of certificate or certificate-related information,
  4. The **Delete** payload indicates one or more SAs that the sender has deleted from its database and that therefore are no longer valid.
  5. The **Vendor ID** payload contains a vendor-defined constant. The constant is used by vendors to identify and recognize remote instances of their implementa-tions. This mechanism allows a vendor to experiment with new features while maintaining backward compatibility.
  6. The **Traffic Selector** payload allows peers to identify packet flows for process-ing by IPsec services.
  7. The **Encrypted** payload contains other payloads in encrypted form. The encrypted payload format is similar to that of ESP. It may include an IV if the encryption algorithm requires it and an ICV if authentication is selected.
  8. The **Configuration** payload is used to exchange configuration information between IKE peers.

4.6.7 Industries

Q18. Write about Intruders.

Ans :

One of the two most publicized threats to security is the intruder (the other is viruses), often referred to as a hacker or cracker. There are three classes of intruders :

- **Masquerader :** An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account.
- **Misfeasor :** A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges.
- **Clandestine user :** An individual who seizes supervisory control of the system and uses this controlto evade auditing and access controls or to suppress audit collection lists the following examples of intrusion :

- Performing a remote root compromise of an e-mail server
- Defacing a Web server
- Guessing and cracking passwords
- Copying a database containing credit card numbers
- Viewing sensitive data, including payroll records and medical information, without authorization
- Running a packet sniffer on a workstation to capture usernames and pass- words
- Using a permission error on an anonymous FTP server to distribute pirated software and music files.
- Dialing into an unsecured modem and gaining internal network access
- Posing as an executive, calling the help desk, resetting the executive's e-mail password, and learning the new password
- Using an unattended, logged-in work station without permission.

4.7 Virus and Firewalls

4.7.1 Intruders

Q19. What are the various types of intruders and explain intrusion techniques ?

Ans :

One of the most publicized attacks to security is the intruder, generally referred to as hacker or cracker. Three classes of intruders are as follows:

- **Masquerader** – an individual who is not authorized to use the computer and who penetrates a system s access controls to exploit a legitimate user s account.
- **Misfeasor** – a legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuse his or her privileges.
- **Clandestine user** – an individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.

Intrusion Techniques

The objective of the intruders is to gain access to a system or to increase the range of privileges accessible on a system. Generally, this requires the intruders to acquire information that should be protected. In most cases, the information is in the form of a user password.

Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it. The password files can be protected in one of the two ways:

- **One way encryption** – the system stores only an encrypted form of user s password. In practice, the system usually performs a one way transformation (not reversible) in which the password is used to generate a key for the encryption function and in which a fixed length output is produced.
- **Access control** – access to the password file is limited to one or a very few accounts.

The following techniques are used for learning passwords.

- Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.
- Exhaustively try all short passwords.
- Try words in the system s online dictionary or a list of likely passwords.
- Collect information about users such as their full names, the name of their spouse and children, pictures in their office and books in their office that are related to hobbies.
- Try user s phone number, social security numbers and room numbers.
- Try all legitimate license plate numbers.
- Use a torjan horse to bypass restriction on access.

- Tap the line between a remote user and the host system.

Two Principle Countermeasures

1. Detection – concerned with learning of an attack, either before or after its success.
2. Prevention – challenging security goal and an uphill bottle at all times.

Q20. What is intrusion detection ? Explain about intrusion detection techniques.

Ans :

Intrusion Detection

Inevitably, the best intrusion prevention system will fail. A system's second line of defense is intrusion detection, and this has been the focus of much research in recent years. This interest is motivated by anumber of considerations, including the following:

i. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficientlytimely to preempt the intruder, the sooner that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved.

ii. An effective intrusion detection system can serve as a deterrent, so acting to pre- vent intrusions.

iii. Intrusion detection enables the collection of information about intrusion tech- niques that can beused to strengthen the intrusion prevention facility.

Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified.

Identifies the following approaches to intrusion detection :

1. **Statistical anomaly detection:** Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a highlevel of confidence whether that behavior is not legitimate user behavior.

   a. **Threshold detection:** This approach involves defining thresholds, independent of user, for thefrequency of occurrence of various events.

   b. **Profile based:** A profile of the activity of each user is developed and used to detect changes in thebehavior of individual accounts.

2. **Rule-based detection :** Involves an attempt to define a set of rules that can be used to decide that agiven behavior is that of an intruder.

   a. **Anomaly detection :** Rules are developed to detect deviation from previous usage patterns.

   b. **Enetration identification :** An expert system approach that searches for suspicious behavior.

Audit Records

A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used :

- **Native audit records:** Virtually all multiuser operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information nor may not contain it in a convenient form.
- **Detection-specific audit records:** A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system. One advantage of suchan approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine.

Each audit record contains the following fields :

- **Subject :** Initiators of actions. A subject is typically a terminal user but might also be a processacting on behalf of users or groups of users. All activity arises through commands issued by subjects. Subjects may be grouped into different access classes, and these classes may overlap.
- **Action :** Operation performed by the subject on or with an object; for example, login, read, performI/O, execute.
- **Object :** Receptors of actions. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures.
- **Exception-Condition :** Denotes which, if any, exception condition is raised on return.
- **Resource-Usage :** A list of quantitative elements in which each element gives the amount used of some resource (e.g., number of lines printed or displayed, number of records read or written, processortime, I/O units used, session elapsed time).
- **Time-Stamp :** Unique time-and-date stamp identifying when the action took place.

Statistical anomaly detection techniques fall into two broad categories: threshold detection and profile-based systems.

Threshold detection involves counting the number of occurrences of aspecific event type over an interval of time. If the count surpasses what is considered a reasonable numberthat one might expect to occur, then intrusion is assumed.

Profile-based anomaly detection focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations. A profile may consist of a set of parameters, sothat deviation on just a single parameter may not be sufficient in itself to signal an alert.

2. Rule-Based Intrusion Detection

Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious.

Rule-based anomaly detection is similar in terms of its approach and strengths to statistical anomaly detection. With the rule-based approach, historical audit records are analyzed to identify usage patterns and to generate automatically rules that describe those patterns. Rules may represent past behavior patterns of users, programs, privileges, time slots, terminals, and so on. Current behavior is then observed, and eachtransaction is matched against the set of rules to determine if it conforms to any historically observed patternof behavior.

Rule-based penetration identification takes a very different approach to intru- sion detection. The key feature of such systems is the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses.

3. Distributed Intrusion Detection

Until recently, work on intrusion detection systems focused on single-system stand- alone facilities. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN or internetwork. Although it is possible to mount a defense by using stand-alone intrusion detection systems on each host, a more effective defense can be achieved by coordination and cooperation among intrusiondetection systems across the network.

Following major issues in the design of a distributed intrusion detection system.

- A distributed intrusion detection system may need to deal with different audit record formats. In aheterogeneous environment, different systems will employ different native audit collection systems and, ifusing intrusion detection, may employ different formats for security-related audit records.
- One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw audit data or summary data must be transmitted across the network.
- Either a centralized or decentralized architecture can be used. With a central- ized architecture, thereis a single central point of collection and analysis of all audit data. This eases the task of correlating incoming reports but creates a potential bottleneck and single point of failure.

4.7.2 Password Management

Q21. Explain about password management and protection.

Ans :

The front line of defense against intruders is the password system. Virtually all multiuser systems requirethat a user provide not only a name or identifier (ID) but also a password. The password serves to authenticate the ID of the individual log- ging on to the system. In turn, the ID provides security in the following ways :

- The ID determines whether the user is authorized to gain access to a system. In some systems, only those who already have an ID filed on the system are allowed to gain access.
- The ID determines the privileges accorded to the user. A few users may have supervisory or "superuser" status that enables them to read files and perform functions that are especially protected by theoperating system. Some systems have guest or anonymous accounts, and users of these accounts have more limited privileges than others.
- The ID is used in what is referred to as discretionary access control. For exam- ple, by listing the IDsof the other users, a user may grant permission to them to read files owned by that user.

Access Control

One way to thwart a password attack is to deny the opponent access to the password file. If the encrypted password portion of the file is accessible only by a privileged user, then the opponent cannotread it without already knowing the password of a privileged user several flaws in thisstrategy:

- Many systems, including most UNIX systems, are susceptible to unanticipated break-ins. Once anattacker has gained access by some means, he or she may wish to obtain a collection of passwords in order touse different accounts for different logon sessions to decrease the risk of detection.
- An accident of protection might render the password file readable, thus com- promising all theaccounts.
- Some of the users have accounts on other machines in other protection domains, and they use the samepassword. Thus, if the passwords could be read by anyone on one machine, a machine in another location mightbe compromised.

Thus, a more effective strategy would be to force users to select passwords that are difficult to guess.

Password Selection Strategies

Four basic techniques are in use :

- User education
- Computer-generated passwords
- Reactive password checking
- Proactive password checking

Users can be told the importance of using hard-to-guess passwords and can be provided with guidelines forselecting strong passwords. This user education strategy is unlikely to succeed at most installations, particularly where there is a large user population or a lot of turnover.

- Computer-generated passwords also have problems. If the passwords are quite random in nature, users willnot be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down..
- A reactive password checking strategy is one in which the system periodically runs its own password crackerto find guessable passwords. The system cancels any passwords that are guessed and notifies the user
- The most promising approach to improved password security is a proactive password checker. In thisscheme, a user is allowed to select his or her own pass- word. However, at the time of selection, the systemchecks to see if the password is allowable and, if not, rejects it. Such checkers are based on the philosophy that, with sufficient guidance from the system, users can select memorable passwords from a fairly large password space that are not likely to be guessed in a dictionary attack.

The first approach is a simple system for rule enforcement. For example, the following rules could be enforced :

- All passwords must be at least eight characters long.
- In the first eight characters, the passwords must include at least one each of uppercase, lowercase,numeric digits, and punctuation marks.

These rules could be coupled with advice to the user. Although this approach is superior to simply educating users, it may not be sufficient to thwart password crackers. This scheme alerts crackers as towhich passwords not to try but may still make it possible to do password cracking.

4.7.3 Viruses and Related Threats

Q22. What are called malicious programs and discuss its types ?

Ans :

Perhaps the most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems.

Malicious Programs

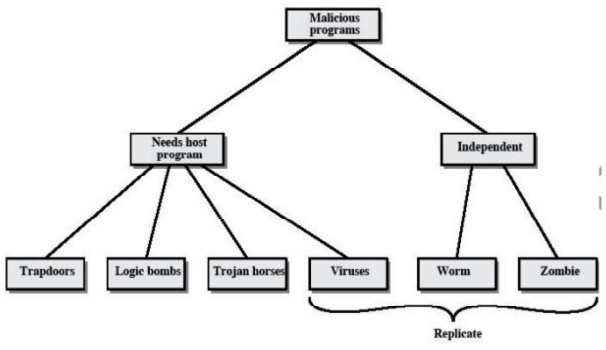


Fig. : Taxonomy of Malicious Programs

Malicious software can be divided into two categories :

- those that need a host program, and
- those that are independent.

The former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples. The latter are self-contained programs that can be scheduled and run by the operating system. Worms and zombie programs are examples.

Name	Description
Virus	Attaches itself to a program and propagates copies of itself to other programs
Worm	Program that propagates copies of itself to other computers
Logic bomb	Triggers action when condition occurs
Trojan horse	Program that contains unexpected additional functionality
Backdoor (Trapdoor)	Program modification that allows unauthorized access to functionality
Exploits	Code specific to a single vulnerability or set of vulnerabilities
Down loaders	Program that installs other items on a machine that Is under attack. Usually, a downloader is sent in an e-mail
Auto – rooter	Malicious hacker tools used to break into a new machines remotely
Kit (Virus generator)	Set of tools for generating new viruses automatically
Spammer programs	Used to send large volumes of unwanted e-mail
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial of service (DoS) attack
Keyloggers	Captured keystrokes on a compromised system
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access
Zombie	Program activated on an infected machine that is activated to launch attacks on other machines

Q23. Write about the nature of viruses.

Ans :

The Nature of Viruses

A virus is a piece of software that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.

A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs.

During its lifetime, a typical virus goes through the following four phases :

- **Dormant phase**: The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.
- **Propagation phase:** The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.
- **Triggering phase:** The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.
- **Execution phase:** The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

Q24. Write about virus structure.

Ans :

Virus Structure

A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.

An infected program begins with the virus code and works as follows.

The first line of code is a jump to the main virus program. The second line is a special marker that is used by the virus to determine whether or not a potential victim program has already been infected with this virus.

When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system.

This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions.

Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.

When this program is invoked, control passes to its virus, which performs the following steps :

1. For each uninfected file P_2 that is found, the virus first compresses that file to produce P'_2, which is shorter than the original program by the size of the virus.
2. A copy of the virus is prepended to the compressed program.
3. The compressed version of the original infected program, P'_1, is uncompressed.
4. The uncompressed original program is executed.

4.7.4 Virus Counter Measures

Q25. What are the various approaches to use to protect from viruses ?

Ans :

Antivirus Approaches

The ideal solution to the threat of viruses is prevention: The next best approach is to be able to do the following :

- **Detection :** Once the infection has occurred, determine that it has occurred and locate the virus.
- **Identification** : Once detection has been achieved, identify the specific virus that has infected a program.
- **Removal** : Once the specific virus has been identified, remove all traces of the virus from the infected program and restore it to its original state. Remove the virus from all infected systems so that the disease cannot spread further.

If detection succeeds but either identification or removal is not possible, then the alternative is to discard the infected program and reload a clean backup version.

There are four generations of antivirus software :

- *First generation*: simple scanners
- *Second generation*: heuristic scanners
- *Third generation* : activity traps
- *Fourth generation*: full-featured protection

- **A first-generation scanner** requires a virus signature to identify a virus.. Such signature-specific scanners are limited to the detection of known viruses. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length.
- **A second-generation scanner** does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable virus infection. One class of such scanners looks for fragments of code that are often associated with viruses.
- **Third-generation programs** are memory-resident programs that identify a virus by its actions rather than its structure in an infected program. Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of viruses. Rather,.
- **Fourth-generation products** are packages consisting of a variety of antivirus techniques used inconjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.

4.7.5 Firewall Design Principles

Q26. What are the fire wall design priciples ?

Ans :

Internet connectivity is no longer an option for most organizations. However, while internet access provides benefits to the organization, it enables the outside world to reach and interact with local network assets. This creates the threat to the organization. While it is possible to equip each workstation and server on the premises network with strong security features, such as intrusion protection, this is not a practical approach. The alternative, increasingly accepted, is the firewall.

The firewall is inserted between the premise network and internet to establish a controlled link and to erect an outer security wall or perimeter. The aim of this perimeter is to protect the premises network from internet based attacks and to provide a single choke point where security and audit can be imposed. The firewall can be a single computer system or a set of two or more systems that cooperate to perform the firewall function.

4.7.6 Types of Firewalls

Q27. Explain various types of firewalls.

Ans :

There are 3 common types of firewalls.

- Packet filters
- Application-level gateways
- Circuit-level gateways

- **Packet filtering router**

A packet filtering router applies a set of rules to each incoming IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions. Filtering rules are based on the information contained in a network packet:

- **Source IP address –** IP address of the system that originated the IP packet. Destination IP address – IP address of the system, the IP is trying to reach. Source and destination transport level address – transport level port number. IP protocol field – defines the transport protocol.
- **Interface –** for a router with three or more ports, which interface of the router the packet come from or which interface of the router the packet is destined for.

  Two default policies are possible :

  - **Default = discard:** That which is not expressly permitted is prohibited.
  - **Default = forward:** That which is not expressly prohibited is permitted.

The default discard policy is the more conservative. Initially everything is blocked, and services must be added on a case-by-case basis. This policy is more visible to users, who are most likely to see the firewall as a hindrance. The default forward policy increases ease of use for end users but provides reduced security.

Advantages of Packet Filter Router

- Simple
- Transparent to users
- Very fast

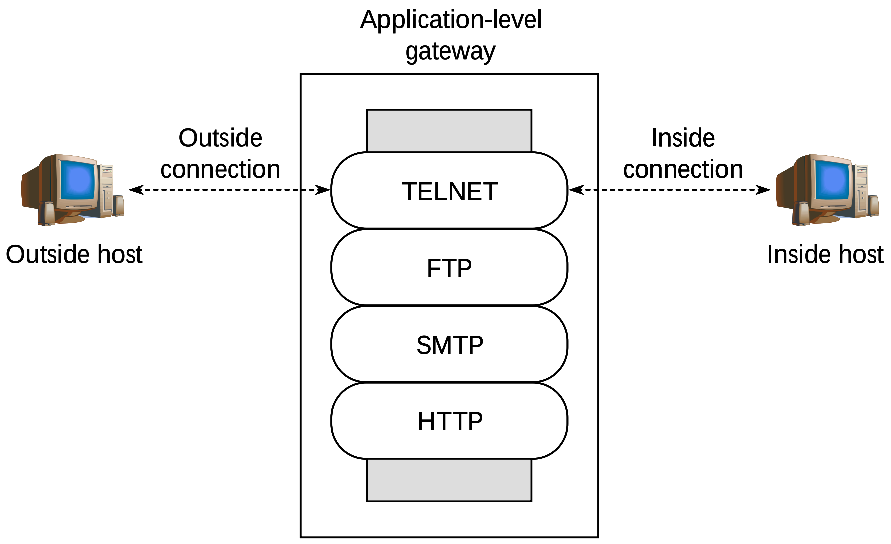
Weakness of Packet Filter Firewalls

- Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application specific vulnerabilities or functions.
- Because of the limited information available to the firewall, the logging functionality present in packet filter firewall is limited.
- It does not support advanced user authentication schemes.
- They are generally vulnerable to attacks such as layer address spoofing.

Application Level Gateway

An Application level gateway, also called a proxy server, acts as a relay of application level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.

Application level gateways tend to be more secure than packet filters. It is easy to log and audit all incoming traffic at the application level. A prime disadvantage is the additional processing overhead on each connection.

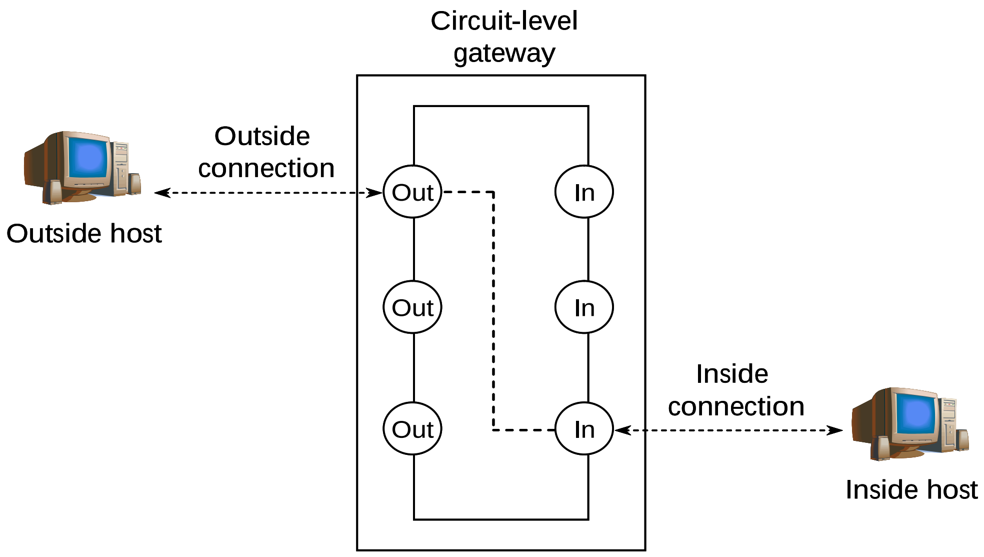


Application-level gateway

Circuit Level Gateway

Circuit level gateway can be a stand-alone system or it can be a specified function performed by an application level gateway for certain applications. A Circuit level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outer host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

A typical use of Circuit level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application level or proxy service on inbound connections and circuit level functions for outbound connections.



Circuit-level gateway

LAB PROGRAMMES
(Implementation Using Java)

Q1. Write a program to generate cipher text and recover the plaintext using.

Ans :

a) Caesar-Cipher text algorithm

```
//A Java Program to illustrate <a href=”#”>Caesar Cipher</a> Technique
class CaesarCipher
{
   // Encrypts text using a shift od s
public static StringBuffer encrypt(String text, int s)
   {
      StringBuffer result= new StringBuffer();

for (int i=0; i<text.length(); i++)
      {
if (Character.isUpperCase(text.charAt(i)))
         {
char ch = (char)(((int)text.charAt(i) +
                              s - 65) % 26 + 65);
result.append(ch);
         }
else
         {
char ch = (char)(((int)text.charAt(i) +
                              s - 97) % 26 + 97);
result.append(ch);
         }
      }
return result;
   }

   // Driver code
public static void main(String[] args)
   {
      String text = “ATTACKATONCE”;
int s = 4;
System.out.println(“Text  : “ + text);
System.out.println(“Shift : “ + s);
System.out.println(“Cipher: “ + encrypt(text, s));
   }
}
```

b) Product Cipher

```
import java.util.*;

class ProductCipher {
public static void main(String args[]) {
System.out.println("Enter the input to be encrypted:");
String substitutionInput = new Scanner(System.in).nextLine();
System.out.println("Enter a number:");
int n = new Scanner(System.in).nextInt();

// Substitution encryption
StringBuffer substitutionOutput = new StringBuffer();
for(int i=0 ; i<substitutionInput.length() ; i++) {
char c = substitutionInput.charAt(i);
substitutionOutput.append((char) (c+5));
}
System.out.println("\nSubstituted text:");
System.out.println(substitutionOutput);

// Transposition encryption
String transpositionInput = substitutionOutput.toString();
int modulus;
if((modulus = transpositionInput.length()%n) != 0) {
modulus = n-modulus;
// 'modulus' is now the number of blanks/padding (X) to be appended
for( ; modulus!=0 ; modulus–) {
transpositionInput += "/";
}
}
StringBuffer transpositionOutput = new StringBuffer();
System.out.println("\nTransposition Matrix:");
for(int i=0 ; i<n ; i++) {
for(int j=0 ; j<transpositionInput.length()/n ; j++) {
char c = transpositionInput.charAt(i+(j*n));
System.out.print(c);
transpositionOutput.append(c);
}
System.out.println();
}
System.out.println("\nFinal encrypted text:");
System.out.println(transpositionOutput);

// Transposition decryption
n = transpositionOutput.length()/n;
StringBuffer transpositionPlaintext = new StringBuffer();
for(int i=0 ; i<n ; i++) {
for(int j=0 ; j<transpositionOutput.length()/n ; j++) {
char c = transpositionOutput.charAt(i+(j*n));
transpositionPlaintext.append(c);
}
}

// Substitution decryption
StringBuffer plaintext = new StringBuffer();
for(int i=0 ; i<transpositionPlaintext.length() ; i++) {
char c = transpositionPlaintext.charAt(i);
plaintext.append((char) (c-5));
}

System.out.println("\nPlaintext:");
System.out.println(plaintext);
}
}

/*
```

Output :

```
Enter the input to be encrypted:
The quick brown fox jumps over the lazy dog.
Enter a number:
7

Substituted text:
Ymj%vznhp%gwt|s%kt}%ozrux%t{jw%ymj%qf#~%itl3

Transposition Matrix:
Yhszjql
mp%rwf3
j%ku%#/
%gtxy~/
vw}%m%/
zt%tji/
n|o{%t/

Final encrypted text:
Yhszjqlmp%rwf3j%ku%#/%gtxy~/vw}%m%/zt%tji/n|o{%t/

Plaintext:
The quick brown fox jumps over the lazy dog.*****

*/
```

Q2. Write a program to generate cipher text and recover the plaintext using

Ans :

a) Play fair cipher

```
importjava.awt.Point;
importjava.util.Scanner;

publicclass PlayfairCipher {
privatestaticchar[][] charTable;
privatestaticPoint[] positions;

publicstaticvoid main(String[] args){
        Scanner sc = new Scanner(System.in);

String key = prompt("Enter an encryption key (min length 6): ", sc, 6);
String txt = prompt("Enter the message: ", sc, 1);
String jti = prompt("Replace J with I? y/n: ", sc, 1);

boolean changeJtoI = jti.equalsIgnoreCase("y");

        createTable(key, changeJtoI);

String enc = encode(prepareText(txt, changeJtoI));

System.out.printf("%nEncoded message:  %n%s%n", enc);
System.out.printf("%nDecoded message:  %n%s%n", decode(enc));
}

privatestaticString prompt(String promptText, Scanner sc, int minLen){
String s;
do{
System.out.print(promptText);
          s = sc.nextLine().trim();
}while(s.length()< minLen);
return s;
}

privatestaticString prepareText(String s, boolean changeJtoI){
        s = s.toUpperCase().replaceAll("[ ^ A-Z]", "");
return changeJtoI ? s.replace("J", "I") : s.replace("Q", "");
}

privatestaticvoid createTable(String key, boolean changeJtoI){
        charTable = newchar[5][5];
        positions = newPoint[26];
```

```
String s = prepareText(key + "ABCDEFGHIJKLMNOPQRSTUVWXYZ", changeJtoI);

int len = s.length();
for(int i = 0, k = 0; i < len; i++){
char c = s.charAt(i);
if(positions[c - 'A'] == null){
            charTable[k / 5][k % 5] = c;
            positions[c - 'A'] = newPoint(k % 5, k / 5);
             k++;
}
}
}

privatestaticString encode(String s){
      StringBuilder sb = new StringBuilder(s);

for(int i = 0; i < sb.length(); i += 2){

if(i == sb.length() - 1)
            sb.append(sb.length() % 2 == 1 ? 'X' : "");

elseif(sb.charAt(i) == sb.charAt(i + 1))
            sb.insert(i + 1, 'X');
}
return codec(sb, 1);
}

privatestaticString decode(String s){
return codec(new StringBuilder(s), 4);
}

privatestaticString codec(StringBuilder text, int direction){
int len = text.length();
for(int i = 0; i < len; i += 2){
char a = text.charAt(i);
char b = text.charAt(i + 1);

int row1 = positions[a - 'A'].y;
int row2 = positions[b - 'A'].y;
int col1 = positions[a - 'A'].x;
int col2 = positions[b - 'A'].x;

if(row1 == row2){
            col1 = (col1 + direction) % 5;
            col2 = (col2 + direction) % 5;
}elseif(col1 == col2){
            row1 = (row1 + direction) % 5;
            row2 = (row2 + direction) % 5;

}else{
int tmp = col1;
            col1 = col2;
            col2 = tmp;
}

         text.setCharAt(i, charTable[row1][col1]);
         text.setCharAt(i + 1, charTable[row2][col2]);
}
return text.toString();
}
}
```

b) Hill cipher

```
package ciphers;
import java.io.*;
import java.lang.*;
public class hillcipher
{
public static void main(String []arg)throws Exception
{
     int k[][]={{17,17,5}, {21,18,21}, {2,2,19}};
     int p[]=new int[300];
     int c[]=new int[300];
     int i=0;
     //System.out.println("enter key");
     BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
     /*for(i=0;i<3;i++)
     {
     for(int j=0;j<3;j++)
     {
     String str=br.readLine();
     k[i][j]=Integer.parseInt(str);
     }
     }*/

     System.out.println("enter plain text");
     String str=br.readLine();
```

```
    for( i=0;i<str.length();i++)
    {
        int c1=str.charAt(i);
        //System.out.println(c1);
        p[i]=(c1)-97;
    }
                    i=0;int zz=0;
        for( int b=0;b<str.length()/3;b++)
        {
        for(int j=0;j<3;j++)
        {
        for(int x=0;x<3;x++)
        {
        c[i]+=k[j][x]*p[x+zz];
        }i++;
        }
        zz+=3;
        }
                    System.out.println("Encrypted Text : ");
        for(int z=0;z<str.length();z++)
        System.out.print((char)((c[z]%26)+97));
}
}
```

Q3. Write a program to generate random numbers using pseudo random number generation algorithm.

Ans :

```
importjava.util.Random;
importjava.util.Scanner;

publicclass Middle_Suqare_Method
{
staticint a[]={1, 10, 100, 1000, 10000, 100000, 1000000, 10000000, 100000000};
staticint middleSquareNumber(int numb, int dig)
{
int sqn = numb*numb, next_num=0;
int trim =(dig/2);
sqn= sqn / a[trim];
for(int i=0; i<dig; i++)
{
        next_num +=(sqn%(a[trim]))*(a[i]);
sqn= sqn/10;
}
return next_num;
}
publicstaticvoid main(String args[])
{
System.out.println("Enter the #-digit random numbers you want: ");
      Scanner sc =newScanner(System.in);
int n = sc.nextInt();

int start=1, end=1;

start= a[n-1];
end= a[n];

Random rand =newRandom();
int number = rand.nextInt(end-start)+start;
System.out.print("The random numbers are:\n"+number+", ");
int new_number=0;
for(int i=0; i<9; i++)
{
number= Middle_Suqare_Method.middleSquareNumber(number, n);
System.out.print(number+", ");
}
System.out.print("...");

sc.close();
}
}
```

Q4. Write a program to implement Poly-Alphabetic Cipher.

Ans :

```
package polyciipher;
import java.util.*;
public class PolyCiipher {

    public static void main(String[] args) {
      int[] j =   new int[100];
         int[] s = new int[100];
         String test="";
         // initialization of the Scanner class,handles input from user,can be found in the java.util.*;
library.
         // we are creating object "in" from the scanner and telling java that this will be System input
          try{
         Scanner in = new Scanner(System.in);
         System.out.println("Enter the plain text(STRING SHOULD BE IN UPPERCASE AND DONT
GIVE SPACE BETWEEN WORDS)::");
```

```
    // next() is a method which  gets the next string of text that a user types on the keyboard
    test = in.nextLine();
    for ( int i = 0; i < test.length(); ++i ) {
   char c = test.charAt( i );// "c" holds the individual character of the string
   s[i] = (int) c-65;

   }
    for(int i=0;i<test.length()-1;i++){
   j[i+1]=s[i];
   }
     System.out.println("Enter the key::");
     int k = Integer.parseInt(in.nextLine());
     j[0]=k;
    System.out.println();
   System.out.println("The position of the character in the cipher text::");
     for(int i=0;i<test.length();i++){
     j[i]=j[i]+s[i];
     j[i]=j[i]%26;
      System.out.print(j[i]);
     }
      System.out.println();
       System.out.println("The cipher text::");
     for(int i=0;i<test.length();i++){
     char c=(char) (j[i]+65);
     System.out.print(c);
     }
       System.out.println();
     }
     catch(Exception er){
    System.out.println("—YOU HAVE TYPE INVALID DATA—");
     }
 }
}
```

Output :

```
run:
Enter a string(STRING SHOULD BE IN ASCII AND DONT GIVE SPACE BETWEEN WORDS)::
ATTACKISTONIGHT
Enter the key::
12

121912192121801171211 4130
MTMTCMSALHBVONA
BUILD SUCCESSFUL (total time: 16 seconds)
```

Q5. Write a program to implement Transposition Cipher and Rail fence Technique.

Ans :

Transposition Cipher (Java)

```
import java.util.*;
class TransCipher
{
public static void main(String args[])
{
Scanner sc=new Scanner(System.in);
System.out.println("Enter the plain text");
String pl=sc.nextLine();
String demo="";
String s="";
int start=0;
for(int i=0;i<pl.length();i++)
{
if(pl.charAt(i)==' ')
{
s=s+pl.substring(start,i);
start=i+1;
}
}
s=s+pl.substring(start);
System.out.print(s);
System.out.println();
//end of space deletion

int k=s.length();
int l=0;
int col=4;
int row=s.length()/col;
char ch[][]=new char[row][col];
for(int i=0;i<row;i++)
{
for(int j=0;j<col;j++)
{
if(l<k)
{
ch[i][j]=s.charAt(l);
l++;
}
else
{
ch[i][j]='#';
}
}
}
//arranged in matrix
char trans[][]=new char[col][row];
for(int i=0;i<row;i++)
{
for(int j=0;j<col;j++)
{
trans[j][i]=ch[i][j];
}
}
for(int i=0;i<col;i++)
{
for(int j=0;j<row;j++)
{
System.out.print(trans[i][j]);
}
}
//display
System.out.println();
}
}
```

Implement Rail Fence Cipher In Java

```
import java.util.*;
class RailFenceBasic{
 int depth;
 String Encryption(String plainText,int depth)throws Exception
 {
  int r=depth,len=plainText.length();
  int c=len/depth;
  char mat[][]=new char[r][c];
  int k=0;

  String cipherText="";

  for(int i=0;i< c;i++)
  {
   for(int j=0;j< r;j++)
   {
    if(k!=len)
     mat[j][i]=plainText.charAt(k++);
    else
     mat[j][i]='X';
   }
  }
  for(int i=0;i< r;i++)
  {
   for(int j=0;j< c;j++)
   {
     cipherText+=mat[i][j];
   }
  }
  return cipherText;
 }

 String Decryption(String cipherText,int depth)throws Exception
 {
  int r=depth,len=cipherText.length();
  int c=len/depth;
  char mat[][]=new char[r][c];
  int k=0;

  String plainText="";

  for(int i=0;i< r;i++)
  {
   for(int j=0;j< c;j++)
   {
     mat[i][j]=cipherText.charAt(k++);
   }
  }
  for(int i=0;i< c;i++)
  {
   for(int j=0;j< r;j++)
   {
     plainText+=mat[j][i];
   }
  }

  return plainText;
 }
}

class RailFence{
 public static void main(String args[])throws Exception
 {
  RailFenceBasic rf=new RailFenceBasic();
                  Scanner scn=new Scanner(System.in);
                  int depth;
```

```
                    String plainText,cipherText,decryptedText;

                    System.out.println("Enter plain text:");
                    plainText=scn.nextLine();

                    System.out.println("Enter depth for Encryption:");
                    depth=scn.nextInt();

  cipherText=rf.Encryption(plainText,depth);
  System.out.println("Encrypted text is:\n"+cipherText);

                    decryptedText=rf.Decryption(cipherText, depth);

  System.out.println("Decrypted text is:\n"+decryptedText);

 }
}
```

Q6. Write a program to implement DES Algorithm.

Ans :

```
import java.security.InvalidKeyException;
 import java.security.NoSuchAlgorithmException;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.NoSuchPaddingException;
import javax.crypto.SecretKey;

import com.sun.mail.util.BASE64DecoderStream;
import com.sun.mail.util.BASE64EncoderStream;

public class EncryptDecryptStringWithDES {

private static Cipher ecipher; private static Cipher dcipher;

        private static SecretKey key;


public static void main(String[] args) {
        try{
                // generate secret key using DES algorithm
                key = KeyGenerator.getInstance("DES").generateKey();

                ecipher = Cipher.getInstance("DES");
                dcipher = Cipher.getInstance("DES");
```

```
                    /// initialize the ciphers with the given key

ecipher.init(Cipher.ENCRYPT_MODE, key);

dcipher.init(Cipher.DECRYPT_MODE, key);
String encrypted = encrypt("This is a classified message!");

String decrypted = decrypt(encrypted);

System.out.println("Decrypted:" + decrypted);
      }

          catch (NoSuchAlgorithmException e) {
                    System.out.println("No Such Algorithm:" + e.getMessageQ);
                    return;
      }
catch (NoSuchPaddingException e) {
System.out.println("No Such Padding:" + e.getMessageQ);
return;
}
catch (InvalidKeyException e) {
System.out.println("Invalid Key:" + e.getMessageQ);
return;
}

}

public static String encrypt(String str) {

try {

// encode the string into a sequence of bytes using the named charset
// storing the result into a new byte array.

byteQ utf8 = str.getBytes("UTF8");

byte[] enc = ecipher.doFinal(utf8);

// encode to base64

enc = BASE64EncoderStream.encode(enc);
return new String(enc);
}

catch (Exception e) {

e.printStackT raceQ;

}
```

```
return null;
}
public static String decrypt(String str) {
try{
// decode with base64 to get bytes
byte[] dec = BASE64DecoderStream.decode(str.getBytesQ ^
byte[] utf8 = dcipher.doFinal(dec);
// create new string based on the specified charset return new String(utf8, “UTF8”);
}
catch (Exception e) {
e.printStackTraceQ;
}
return null;
}
}
```

Q7. Write a program to implement AES Algorithm

Ans :

```
package nomad;

import java.security.*;
import java.security.spec.InvalidKeySpecException;
import javax.crypto.*;
import sun.misc.*;

publicclass AESencrp {

privatestaticfinalString ALGO = “AES”;
privatestaticfinal byte[] keyValue =
new byte[] { ‘T’, ‘h’, ‘e’, ‘B’, ‘e’, ‘s’, ‘t’,
‘S’, ‘e’, ‘c’, ‘r’,’e’, ‘t’, ‘K’, ‘e’, ‘y’ };

publicstaticString encrypt(String Data) throwsException {
Key key = generateKey();
      Cipher c = Cipher.getInstance(ALGO);
c.init(Cipher.ENCRYPT_MODE, key);
byte[] encVal = c.doFinal(Data.getBytes());
String encryptedValue = newBASE64Encoder().encode(encVal);
return encryptedValue;
   }

publicstaticString decrypt(String encryptedData) throwsException {
Key key = generateKey();
      Cipher c = Cipher.getInstance(ALGO);
c.init(Cipher.DECRYPT_MODE, key);
byte[] decordedValue = new BASE64Decoder().decodeBuffer(encryptedData);
byte[] decValue = c.doFinal(decordedValue);
String decryptedValue = newString(decValue);
return decryptedValue;
   }
privatestaticKey generateKey() throwsException {
Key key = newSecretKeySpec(keyValue, ALGO);
return key;
}
}
publicclass Checker {

publicstatic void main(String[] args) throwsException {

String password = "mypassword";
String passwordEnc = AESencrp.encrypt(password);
String passwordDec = AESencrp.decrypt(passwordEnc);

System.out.println("Plain Text : " + password);
System.out.println("Encrypted Text : " + passwordEnc);
System.out.println("Decrypted Text : " + passwordDec);
   }
}
```

Q8. Write a program to implement RSA public key encryption algorithm.

Ans :

```
import java.math.BigInteger;
import java.security.SecureRandom;
publicclass RSA {
private BigInteger n, d, e;

privateint bitlen = 1024;

  /** Create an instance that can encrypt using someone elses public key. */
public RSA(BigInteger newn, BigInteger newe) {
   n = newn;
   e = newe;
  }

  /** Create an instance that can both encrypt and decrypt. */
public RSA(int bits) {
bitlen = bits;
   SecureRandom r = newSecureRandom();
   BigInteger p = newBigInteger(bitlen / 2, 100, r);
```

```
    BigInteger q = newBigInteger(bitlen / 2, 100, r);
    n = p.multiply(q);
    BigInteger m = (p.subtract(BigInteger.ONE)).multiply(q
        .subtract(BigInteger.ONE));
   e = newBigInteger("3");
while (m.gcd(e).intValue() > 1) {
      e = e.add(new BigInteger("2"));
    }
    d = e.modInverse(m);
  }

  /** Encrypt the given plaintext message. */
publicsynchronized String encrypt(String message) {
return (new BigInteger(message.getBytes())).modPow(e, n).toString();
  }

  /** Encrypt the given plaintext message. */
publicsynchronized BigInteger encrypt(BigInteger message) {
return message.modPow(e, n);
  }

  /** Decrypt the given ciphertext message. */
publicsynchronized String decrypt(String message) {
returnnew String((new BigInteger(message)).modPow(d, n).toByteArray());
  }

  /** Decrypt the given ciphertext message. */
publicsynchronized BigInteger decrypt(BigInteger message) {
return message.modPow(d, n);
  }

  /** Generate a new public and private key set. */
publicsynchronizedvoid generateKeys() {
    SecureRandom r = newSecureRandom();
    BigInteger p = newBigInteger(bitlen / 2, 100, r);
    BigInteger q = newBigInteger(bitlen / 2, 100, r);
    n = p.multiply(q);
    BigInteger m = (p.subtract(BigInteger.ONE)).multiply(q
        .subtract(BigInteger.ONE));
   e = newBigInteger("3");
while (m.gcd(e).intValue() > 1) {
      e = e.add(new BigInteger("2"));
    }
    d = e.modInverse(m);
  }
  /** Return the modulus. */
publicsynchronized BigInteger getN() {
return n;
  }

  /** Return the public key. */
publicsynchronized BigInteger getE() {
return e;
  }

  /** Trivial test program. */
publicstaticvoid main(String[] args) {
    RSA rsa = newRSA(1024);

    String text1 = "Yellow and Black Border Collies";
System.out.println("Plaintext: " + text1);
    BigInteger plaintext = newBigInteger(text1.getBytes());

    BigInteger ciphertext = rsa.encrypt(plaintext);
System.out.println("Ciphertext: " + ciphertext);
plaintext = rsa.decrypt(ciphertext);

    String text2 = newString(plaintext.toByteArray());
System.out.println("Plaintext: " + text2);
  }
}
```

Q9. Write a program to implement Diffie- Helman Key Exchange Algorithm.

Ans :

```
import java.util.*;
class DiffieHellman
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
System.out.println("Enter the value of Xa & Xb");
int Xa=sc.nextInt();
int Xb=sc.nextInt();
System.out.println("Enter a Prime no. p");
int p=sc.nextInt();
System.out.println("Enter Primitive Root a, such that a<p");
int a=sc.nextInt();
int Ya=(int)((Math.pow(a,Xa))%p);
int Yb=(int)((Math.pow(a,Xb))%p);
int Ka=(int)((Math.pow(Yb,Xa))%p);
int Kb=(int)((Math.pow(Ya,Xb))%p);
if(Ka= =Kb)
{
System.out.println("Transmission  successful");
}
else
{
System.out.println("Transmission  failed");
}
}
}
```

Q10. Write a program to implement SHA-1 algorithm.

Ans :

```
import java.io.FileInputStream;
import java.security.MessageDigest;

publicclassSHACheckSumExample
{
publicstaticvoidmain(String[] args)throws Exception
{
      MessageDigest md =MessageDigest.getInstance("SHA-256");
       FileInputStream fis =newFileInputStream("c:\\loging.log");

byte[] dataBytes =newbyte[1024];

int nread =0;
while((nread = fis.read(dataBytes))!=-1){
md.update(dataBytes,0, nread);
};
byte[] mdbytes = md.digest();

//convert the byte to hex format method 1
      StringBuffer sb =newStringBuffer();
for(int i =0; i < mdbytes.length; i++){
sb.append(Integer.toString((mdbytes[i]&0xff)+0x100,16).substring(1));
}

System.out.println("Hex format : "+ sb.toString());

//convert the byte to hex format method 2
      StringBuffer hexString =newStringBuffer();
            for(int i=0;i<mdbytes.length;i++){
            hexString.append(Integer.toHexString(0xFF& mdbytes[i]));
            }
```

```
          System.out.println("Hex format : "+ hexString.toString());
}
}
```

Output

```
Hex format: 21a57f2fe765e1ae4a8bf15d73fc1bf2a533f547f2343d12a499d9c0592044d4
Hex format: 21a57f2fe765e1ae4a8bf15d73fc1bf2a533f547f2343d12a499d9c0592044d4
```

Q11. Write a program to implement MD5 algorithm.

Ans :

```
import java.io.FileInputStream;
import java.io.UnsupportedEncodingException;
import java.math.BigInteger;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;

public class MD5 {
     public static String getMD5(String input) {
          try {
               MessageDigest md = MessageDigest.getInstance("MD5");
               byte[] messageDigest = md.digest(input.getBytes());
               BigInteger number = new BigInteger(1, messageDigest);
               String hashtext = number.toString(16);
               // Now we need to zero pad it if you actually want the full 32 chars.
               while (hashtext.length() < 32) {
                    hashtext = "0" + hashtext;
                }
               return hashtext;
          }
          catch (NoSuchAlgorithmException e) {
               throw new RuntimeException(e);
          }
     }

     public static void main(String[] args) throws NoSuchAlgorithmException {
          System.out.println(getMD5("Javarmi.com"));
     }
}
```

Q12. Write a program to Implement the Signature Scheme Digital Signature Standard.

Ans :

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectOutputStream;
import java.nio.file.Files;
import java.security.InvalidKeyException;
import java.security.KeyFactory;
import java.security.PrivateKey;
import java.security.Signature;
import java.security.spec.PKCS8EncodedKeySpec;
import java.util.ArrayList;
import java.util.List;

import javax.swing.JOptionPane;

public class Message {
        private List<byte[]> list;

        //The constructor of Message class builds the list that will be written to the file.
        //The list consists of the message and the signature.
        public Message(String data, String keyFile) throws InvalidKeyException, Exception {
                list = new ArrayList<byte[]>();
                list.add(data.getBytes());
                list.add(sign(data, keyFile));
        }

        //The method that signs the data using the private key that is stored in keyFile path
        public byte[] sign(String data, String keyFile) throws InvalidKeyException, Exception{
                Signature rsa = Signature.getInstance("SHA1withRSA");
                rsa.initSign(getPrivate(keyFile));
                rsa.update(data.getBytes());
                return rsa.sign();
        }

        //Method to retrieve the Private Key from a file
        public PrivateKey getPrivate(String filename) throws Exception {
                byte[] keyBytes = Files.readAllBytes(new File(filename).toPath());
                PKCS8EncodedKeySpec spec = new PKCS8EncodedKeySpec(keyBytes);
                KeyFactory kf = KeyFactory.getInstance("RSA");
                return kf.generatePrivate(spec);
        }

        //Method to write the List of byte[] to a file
        private void writeToFile(String filename) throws FileNotFoundException, IOException {
                File f = new File(filename);
                f.getParentFile().mkdirs();
                ObjectOutputStream out = new ObjectOutputStream(new
                                FileOutputStream(filename));
```

```
            out.writeObject(list);
                        out.close();
                        System.out.println("Your file is ready.");
            }

            public static void main(String[] args) throws InvalidKeyException, IOException, Exception{
                        String data = JOptionPane.showInputDialog("Type your message here");
                        new Message(data, "MyKeys/privateKey").writeToFile("MyData/SignedData.txt");
            }
}
```

Q13. Write a program to retrieve the data from the database and encrypt them using any encryption algorithm.

Ans :

```
import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.xml.bind.DatatypeConverter;

/**
 * This example program shows how AES encryption and decryption can be done in Java.
 * Please note that secret key and encrypted text is unreadable binary and hence
 * in the following program we display it in hexadecimal format of the underlying bytes.
 * @author Jayson
 */
public class AESEncryption {

   /**
    * 1. Generate a plain text for encryption
    * 2. Get a secret key (printed in hexadecimal form). In actual use this must
    * by encrypted and kept safe. The same key is required for decryption.
    * 3.
    */
public static void main(String[] args) throws Exception {
      String plainText = "Hello World";
       SecretKey secKey = getSecretEncryptionKey();
byte[] cipherText = encryptText(plainText, secKey);
      String decryptedText = decryptText(cipherText, secKey);

System.out.println("Original Text:" + plainText);
System.out.println("AES Key (Hex Form):"+bytesToHex(secKey.getEncoded()));
System.out.println("Encrypted Text (Hex Form):"+bytesToHex(cipherText));
System.out.println("Descrypted Text:"+decryptedText);

   }
   /**
* gets the AES encryption key. In your actual programs, this should be safely
     * stored.
     * @return
     * @throws Exception
     */
public static SecretKey getSecretEncryptionKey() throws Exception{
        KeyGenerator generator = KeyGenerator.getInstance("AES");
generator.init(128); // The AES key size in number of bits
        SecretKey secKey = generator.generateKey();
return secKey;
   }

   /**
     * Encrypts plainText in AES using the secret key
     * @param plainText
     * @param secKey
     * @return
     * @throws Exception
     */
public static byte[] encryptText(String plainText,SecretKey secKey) throws Exception{
                            // AES defaults to AES/ECB/PKCS5Padding in Java 7
        Cipher aesCipher = Cipher.getInstance("AES");
aesCipher.init(Cipher.ENCRYPT_MODE, secKey);
byte[] byteCipherText = aesCipher.doFinal(plainText.getBytes());
return byteCipherText;
   }

   /**
     * Decrypts encrypted byte array using the key used for encryption.
     * @param byteCipherText
     * @param secKey
     * @return
     * @throws Exception
     */
public static String decryptText(byte[] byteCipherText, SecretKey secKey) throws Exception {
                            // AES defaults to AES/ECB/PKCS5Padding in Java 7
        Cipher aesCipher = Cipher.getInstance("AES");
aesCipher.init(Cipher.DECRYPT_MODE, secKey);
byte[] bytePlainText = aesCipher.doFinal(byteCipherText);
return new String(bytePlainText);
   }
   /**
    * Convert a binary byte array into readable hex form
    * @param hash
    * @return
    */
private static String  bytesToHex(byte[] hash) {
return  DatatypeConverter.printHexBinary(hash);
   }
}
```

Q14. Generation of public key and private keys.

Ans :

```
import java.security.KeyPairGenerator;
import java.security.KeyPair;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.KeyFactory;
import java.security.spec.EncodedKeySpec;
import java.security.spec.PKCS8EncodedKeySpec;
import java.security.spec.X509EncodedKeySpec;
import java.security.spec.InvalidKeySpecException;
import java.security.NoSuchAlgorithmException;

public class GeneratePublicPrivateKeys {

    private static void generateKeys(   String keyAlgorithm
                                        , int      numBits) {

        try {

            // Get the public/private key pair
            KeyPairGenerator keyGen = KeyPairGenerator.getInstance(keyAlgorithm);
            keyGen.initialize(numBits);
            KeyPair keyPair = keyGen.genKeyPair();
            PrivateKey privateKey = keyPair.getPrivate();
            PublicKey   publicKey   = keyPair.getPublic();

            System.out.println(
                    "\n" +
                    "Generating key/value pair using " +
                    privateKey.getAlgorithm() +
                    " algorithm");
            // Get the bytes of the public and private keys
            byte[] privateKeyBytes = privateKey.getEncoded();
            byte[] publicKeyBytes   = publicKey.getEncoded();

            // Get the formats of the encoded bytes
            String formatPrivate = privateKey.getFormat(); // PKCS#8
            String formatPublic   = publicKey.getFormat(); // X.509

            System.out.println("   Private Key Format : " + formatPrivate);
            System.out.println("   Public Key Format   : " + formatPublic);

            // The bytes can be converted back to public and private key objects
            KeyFactory keyFactory = KeyFactory.getInstance(keyAlgorithm);
            EncodedKeySpec privateKeySpec = new PKCS8EncodedKeySpec(privateKeyBytes);
            PrivateKey privateKey2 = keyFactory.generatePrivate(privateKeySpec);

            EncodedKeySpec publicKeySpec = new X509EncodedKeySpec(publicKeyBytes);
            PublicKey publicKey2 = keyFactory.generatePublic(publicKeySpec);

            // The original and new keys are the same
            System.out.println(
                "   Are both private keys equal? " + privateKey.equals(privateKey2));

            System.out.println(
                "   Are both public keys equal? " + publicKey.equals(publicKey2));

        } catch (InvalidKeySpecException specException) {

            System.out.println("Exception");
            System.out.println("Invalid Key Spec Exception");

        } catch (NoSuchAlgorithmException e) {

            System.out.println("Exception");
            System.out.println("No such algorithm: " + keyAlgorithm);
        }
    }

    public static void main(String[] args) {

        // Generate a 1024-bit Digital Signature Algorithm (DSA) key pair
        generateKeys("DSA", 1024);

        // Generate a 576-bit DH key pair
        generateKeys("DH", 576);
        // Generate a 1024-bit RSA key pair
        generateKeys("RSA", 1024);
    }

}
```

15. Write a program to write the contents in the file in encrypted manner and read them in decrypted manner using any algorithm.

Ans :

```
package com.javapapers.java.security;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.util.Random;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.PBEParameterSpec;

publicclassFileEncryption{

        publicstaticvoid main(String[] args)throwsException{

                FileInputStream inFile =newFileInputStream("plainfile.txt");
                FileOutputStream outFile =newFileOutputStream("plainfile.des");

                String password = "javapapers";
                PBEKeySpec pbeKeySpec =newPBEKeySpec(password.toCharArray());
                SecretKeyFactory secretKeyFactory =SecretKeyFactory
                                .getInstance("PBEWithMD5AndTripleDES");
                SecretKey secretKey =secretKeyFactory.generateSecret(pbeKeySpec);
                byte[] salt =newbyte[8];
                Random random =newRandom();
                random.nextBytes(salt);
                PBEParameterSpec pbeParameterSpec =newPBEParameterSpec(salt,100);
                Cipher cipher =Cipher.getInstance("PBEWithMD5AndTripleDES");
                cipher.init(Cipher.ENCRYPT_MODE, secretKey, pbeParameterSpec);
                outFile.write(salt);

                byte[] input =newbyte[64];
                int bytesRead;
```

```
                    while((bytesRead = inFile.read(input))!=-1){
                              byte[] output = cipher.update(input,0, bytesRead);
                              if(output !=null)
                                        outFile.write(output);
                    }

                    byte[] output = cipher.doFinal();
                    if(output !=null)
                              outFile.write(output);

                    inFile.close();
                    outFile.flush();
                    outFile.close();
          }
}
```

FileDecryption.java

```
package com.javapapers.java.security;

import java.io.FileInputStream;
import java.io.FileOutputStream;

import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.SecretKeyFactory;
import javax.crypto.spec.PBEKeySpec;
import javax.crypto.spec.PBEParameterSpec;
publicclassFileDecryption{
          publicstaticvoid main(String[] args)throwsException{
                    String password =”javapapers”;
                    PBEKeySpec pbeKeySpec =newPBEKeySpec(password.toCharArray());
                    SecretKeyFactory secretKeyFactory =SecretKeyFactory
                                        .getInstance(“PBEWithMD5AndTripleDES”);
                    SecretKey secretKey =secretKeyFactory.generateSecret(pbeKeySpec);

                    FileInputStream fis =newFileInputStream(“plainfile.des”);
                    byte[] salt =newbyte[8];
                    fis.read(salt);

                    PBEParameterSpec pbeParameterSpec =newPBEParameterSpec(salt,100);

                    Cipher cipher =Cipher.getInstance(“PBEWithMD5AndTripleDES”);
                    cipher.init(Cipher.DECRYPT_MODE, secretKey, pbeParameterSpec);
                    FileOutputStream fos =newFileOutputStream(“plainfile_decrypted.txt”);
                    byte[] in =newbyte[64];
                    int read;
                    while((read = fis.read(in))!=-1){
                              byte[] output = cipher.update(in,0, read);
                              if(output !=null)
                                        fos.write(output);
                    }

                    byte[] output = cipher.doFinal();
                    if(output !=null)
                              fos.write(output);

                    fis.close();
                    fos.flush();
                    fos.close();
          }
}
```

Q16. Write a program to implement FEM (Fast Exponentiation Method) for find congruential values used in RSA algorithms Ex: 577mod 45.

Ans :

```
public static int modpow(int value , int power, int mod){
int e = 1;

for (int i = 0; i < power; i++) {
      e = ((e * value) % mod);

   }

return e;
}//modpow

/*

((value ^ power)%mod)

call as modpow(value, power, mod);

*/
```

FACULTY OF SCIENCE

M.Sc. II Year III - Semester Examination

NETWORK SECURITY

Time : 3 Hours] **ANSWERS TO MODEL PAPER - I** [Max. Marks : 80

PART - A (8 × 4 = 32)

Answer all questions

Each Question Carries Equal Marks **ANSWERS :**

1.	What is computer security?	**(Refer to Q.No.1, Unit -1)**
2.	What is Padding in Block Cipher ?	**(Refer to Q.No.14, Unit -1)**
3.	Explain about the features of AES.	**(Refer to Q.No.1, Unit -2)**
4.	What are stream ciphers ?	**(Refer to Q.No.10, Unit -2)**
5.	Write briefly about Message Authentication.	**(Refer to Q No.1, Unit -3)**
6.	Write shortly about Message Authentication Code.	**(Refer to Q.No.6, Unit -3)**
7.	Explain SSL Record Protocol	**(Refer to Q.No.3, Unit -4)**
8.	Write about transport mode and tunnel mode	**(Refer to Q No.13, Unit -4)**

PART - B (4 × 12 = 48)

Answer any four from the following

9.	What are rotor machines? Explain	**(Refer to Q No.11, Unit - I)**
10.	What is cipher block chaining mode? Explain.	**(Refer to Q No.21, Unit 1)**
11.	Explain AES Key Expansion algorithm. Illustrate AES with an example.	**(Refer to Q No.4,5, Unit -2)**
12.	Write about x.509 certificates.	**(Refer to Q No.16, Unit -2)**
13.	What are the message authentication requirements ?	**(Refer to Q No.5, Unit -3)**
14.	Explain briefly about digital signatures.	**(Refer to Q No.12, Unit -3)**
15.	Explain about ESP(Encapsulaitng Security Payload)	**(Refer to Q No.15, Unit -4)**
16.	Explain various types of firewalls.	**(Refer to Q No.27, Unit -4)**

FACULTY OF SCIENCE

M.Sc. II Year III - Semester Examination

NETWORK SECURITY

Time : 3 Hours] **ANSWERS TO MODEL PAPER - II** [Max. Marks : 80

PART - A (8 × 4 = 32)

Answer all questions

Each Question Carries Equal Marks **ANSWERS :**

1.	Explain about various security attacks in network.	**(Refer to Q.No.3, Unit -1)**
2.	What is block cipher?	**(Refer to Q.No.13, Unit -1)**
3.	Explain about the structure of AES.	**(Refer to Q.No.2, Unit -2)**
4.	What are called Round Keys?	**(Refer to Q.No.3, Unit -2)**
5.	What are Digital Signatures ?	**(Refer to Q.No.1, Unit -3)**
6.	What is Message Encryption ?	**(Refer to Q.No.6, Unit -3)**
7.	Discuss about Alter Protocol.	**(Refer to Q.No.4, Unit -4)**
8.	Explain IP security services.	**(Refer to Q.No.12, Unit -4)**

PART - B (4 × 12 = 48)

Answer any four from the following

9.	Explain about DES algorithm.	**(Refer to Q.No.5, Unit -1)**
10.	What is Electronic code Book ?write a note on it.	**(Refer to Q.No.20, Unit -1)**
11.	What are stream ciphers ? Explain about it.	**(Refer to Q.No.10, Unit -2)**
12.	Explain briefly RSA Algorithm.	**(Refer to Q.No.13, Unit -2)**
13.	Explain about MD5 algorithm.	**(Refer to Q.No.4, Unit -3)**
14.	Write about various attacks on MACs.	**(Refer to Q.No.8, Unit -3)**
15.	Explain briefly about HHTPS protocol.	**(Refer to Q.No.6, Unit -4)**
16.	Describe IP Security Architecture.	**(Refer to Q.No.14, Unit -4)**

FACULTY OF SCIENCE

M.Sc. II Year III - Semester Examination

NETWORK SECURITY

Time : 3 Hours] **ANSWERS TO MODEL PAPER - III** [Max. Marks : 80

PART - A (8 × 4 = 32)

Answer all questions

Each Question Carries Equal Marks **ANSWERS :**

1. Explain about various security services in network. **(Refer to Q.No.4, Unit -1)**
2. Write the strength of DES. **(Refer to Q.No.17, Unit -1)**
3. What is the use of Random Numbers ? **(Refer to Q.No.7, Unit -2)**
4. What are Public-Key Certificates ? **(Refer to Q.No.15, Unit -2)**
5. Write shortly about Message Authentication Code. **(Refer to Q.No.6, Unit -3)**
6. Write about Digital Signature Requirements. **(Refer to Q.No.12, Unit -3)**
7. Explain about Handshake protocol. **(Refer to Q.No.5, Unit -4)**
8. What is called port forwarding ? **(Refer to Q.No.7, Unit -4)**

PART - B (4 × 12 = 48)

Answer any four from the following

9. Write about the symmetric cipher model. **(Refer to Q.No.8, Unit -1)**
10. What is Output Feedback Mode? Explain **(Refer to Q.No.23, Unit -1)**
11. Explain, how to generate a pseudo random number using a block cipher. **(Refer to Q.No.9, Unit -2)**
12. Write about Diffie-Helman Algorithm. **(Refer to Q.No.17, Unit -2)**
13. Write about various attacks on MACs. **(Refer to Q.No.8, Unit -3)**
14. Explain about HMAC algorithm. **(Refer to Q.No.9, Unit -3)**
15. What is intrusion detection ? Explain about intrusion detection techniques. **(Refer to Q.No.20, Unit -4)**
16. Explain about password management and protection. **(Refer to Q.No.21, Unit -4)**