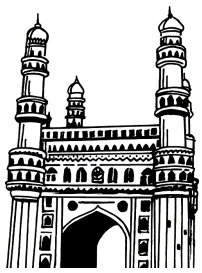
Rahul's ✔
Topper's Voice

AS PER
CBCS SYLLABUS



M.C.A

II Year III Sem

Latest 2022 Edition

NETWORK SECURITY

- ☞ **Study Manual**
- ☞ **Important Questions**
- ☞ **Solved Model Papers**

~~₹199/-~~
₹159/-

- by -

WELL EXPERIENCED LECTURER

Rahul Publications™
Hyderabad. Ph : 66550071, 9391018098

M.C.A

II Year III Sem

NETWORK SECURITY

Inspite of many efforts taken to present this book without errors, some errors might have crept in. Therefore we do not take any legal responsibility for such errors and omissions. However, if they are brought to our notice, they will be corrected in the next edition.

Price ` : ~~***199-00***~~
: 159-00

Sole Distributors :

☎ **: 66550071, Cell : 9391018098**

VASU BOOK CENTRE

Shop No. 2, Beside Gokul Chat, Koti, Hyderabad.

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

NETWORK SECURITY

CONTENTS

STUDY MANUAL

SOLVED MODEL PAPERS

SYLLABUS

UNIT - I

Introduction: Attributes of Security, Integrity, Authenticity, Non-repudiation, Confidentiality Authorization, Anonymity, Types of Attacks, DoS, IP Spoofing, Replay, Man-in-the-Middle attacks General Threats to Computer Network, Worms, Viruses, - Trojans

UNIT - II

Secret Key Cryptography : DES, Triple DES, AES, Key distribution, Attacks

Public Key Cryptography: RSA, ECC, Key Exchange (Diffie-Hellman), Java Cryptography Extensions, Attacks

UNIT - III

Integrity, Authentication and Non-Repudiation : Hash Function (MD5, SHA5), Message Authentication Code (MAC), Digital Signature (RSA, DSA Signatures), Biometric Authentication.

UNIT - IV

PKI Interface: Digital Certificates, Certifying Authorities, POP Key Interface, System Security using Firewalls and VPN's.

Smart Cards: Application Security using Smart Cards, Zero Knowledge Protocols and their use in Smart Cards, Attacks on Smart Cards

UNIT - V

Applications: Kerberos, Web Security Protocols (SSL), IPSec, Electronic Payments, E-cash, Secure Electronic Transaction (SET), Micro Payments, Case Studies of Enterprise Security (.NET and J2EE)

Contents

UNIT - I

UNIT - II

UNIT - III

Important Questions

UNIT - I

1. Describe the basic terminology used in network security.

Ans :

Refer Unit-I, Q.No. 2

2. What are the various attributes of security?

Ans :

Refer Unit-I, Q.No. 3

3. Explain different types of security attacks.

Ans :

Refer Unit-I, Q.No. 4

4. Explain the model for network security.

Ans :

Refer Unit-I, Q.No. 5

5. What is Denial of service attack?

Ans :

Refer Unit-I, Q.No. 6

6. Explain General Threats to Computer Network.

Ans :

Refer Unit-I, Q.No. 10

7. Explain the classification of virus.

Ans :

Refer Unit-I, Q.No. 11

UNIT - II

1. Describe the concept of DES algorithm.

Ans :

Refer Unit-II, Q.No. 1

2. Explain the strength of DES.

Ans :

Refer Unit-II, Q.No. 3

3. Define AES. Explain the features of AES.

Ans :

Refer Unit-II, Q.No. 5

4. Explain about AES encryption round key.

Ans :

Refer Unit-II, Q.No. 7

5. Illustrate AES with an example.

Ans :

Refer Unit-II, Q.No. 9

6. Explain about Symmetric Key distribution using Asymmetric Encryption.

Ans :

Refer Unit-II, Q.No. 11

7. Explain various principles of public key cryptography.

Ans :

Refer Unit-II, Q.No. 12

8. Explain about Elliptic Curve Cryptography.

Ans :

Refer Unit-II, Q.No. 14

9. Briefly explain about Diffie - Hellman algorithm.

Ans :

Refer Unit-II, Q.No. 15

UNIT - III

1. Explain the concept of Hash Function.

Ans :

Refer Unit-III, Q.No. 1

2. Describe the concept of message degest algorithm.

Ans :

Refer Unit-III, Q.No. 2

3. What is secure hash algorithm? Explain.

Ans :

Refer Unit-III, Q.No. 3

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

Ans :

Refer Unit-III, Q.No. 6

5. Explain the concept of Digital Signature.

Ans :

Refer Unit-III, Q.No. 9

6. Explain DSA algorithm.

Ans :

Refer Unit-III, Q.No. 11

7. Define Biometric Authentication. Explain its types.

Ans :

Refer Unit-III, Q.No. 12

8. Explain the evolution of biometric verification.

Ans :

Refer Unit-III, Q.No. 13

UNIT - IV

1. What do you mean by PKI interface.

Ans :

Refer Unit-IV, Q.No. 1

2. Explain the concept of Digital Certi- ficate and CA.

Ans :

Refer Unit-IV, Q.No. 2

3. Explain different types of firewalls.

Ans :

Refer Unit-IV, Q.No. 7

4. Explain the concept of firewall configuration.

Ans :

Refer Unit-IV, Q.No. 8

5. Explain different types of VPN's

Ans :

Refer Unit-IV, Q.No. 11

6. Explain the concept of zero knowledge protocol.

Ans :

Refer Unit-IV, Q.No. 15

7. Explain different types of attacks on Smart cards.

Ans :

Refer Unit-IV, Q.No. 17

UNIT - V

1. Explain about the various keys used with kerberos.

Ans :

Refer Unit-V, Q.No. 2

2. What are the various types of tickets with kerberos V5?

Ans :

Refer Unit-V, Q.No. 6

3. Explain the concept of SSL

Ans :

Refer Unit-V, Q.No. 9

4. Explain the concept of IPSec.

Ans :

Refer Unit-V, Q.No. 10

5. Explain the concept of Secure Electronic Transaction (SET).

Ans :

Refer Unit-V, Q.No. 15

6. Describe the concept of mixro payments.

Ans :

Refer Unit-V, Q.No. 16

7. What are the elements of .NET?

Ans :

Refer Unit-V, Q.No. 17

8. Explain the architecture of J2EE.

Ans :

Refer Unit-V, Q.No. 20

UNIT I

INTRODUCTION:

Attributes of Security. Integrity, Authenticity, Non-repudiation, Confidentiality Authorization, Anonymity, Types of Attacks, DoS, IP Spoofing, Replay, Man-in-the-Middle attacks General Threats to Computer Network, Worms, Viruses, -Trojans

1.1 Network Security

Q1. What is Network Security?

Ans :

Meaning

Network security is the process of taking preventative measures to protect the underlying networking infrastructure from unauthorized access, misuse, malfunction, modification, destruction or improper disclosure.

The Internet has undoubtedly become a huge part of our lives. Many people in today's generation rely on the Internet for many of their professional, social and personal activities. But are you sure your network is secure?

There are many people who attempt to damage our Internet-connected computers, violate our privacy and make it impossible to the Internet services. Given the frequency and variety of existing attacks as well as the threat of new and more destructive future attacks, network security has become a central topic in the field of cybersecurity. Implementing network security measures allows computers, users and programs to perform their permitted critical functions within a secure environment.

(i) **Computer Security:** generic name for the collection of tools designed to protect data and to thwart hackers

(ii) **Network Security:** measures to protect data during their transmission

(iii) **Internet Security:** measures to protect data during their transmission over a collection of interconnected networks

Q2. Explain the basic concepts of network security.

(OR)

Describe the basic terminology used in network security.

Ans : **(Imp.)**

(i) **Cryptography**

The art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form

(ii) **Plaintext:** The original intelligible message

(iii) **Cipher text:** The transformed message

(iv) **Cipher:** An algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods

(v) **Key:** Some critical information used by the cipher, known only to the sender& receiver

(vi) **Encipher(encode):** The process of converting plaintext to cipher text using a cipher and a key

(vii) **Decipher (decode):** the process of converting cipher text back into plaintext using a cipher and a key

(viii) **Cryptanalysis:** The study of principles and methods of transforming an unintelligible message back into an intelligible message without knowledge of the key. Also called code breaking.

(ix) Cryptology: Both cryptography and cryptanalysis

(x) Code: An algorithm for transforming an intelligible message into an unintelligible one using a code-book

1.2 Attributes of Security

1.2.1 Integrity, Authenticity, Non-repudiation, Confidentiality Authorization, Anonymity

Q3. What are the various attributes of security?

Ans : **(Imp.)**

1. Authentication

The authentication service is concerned with assuring that a communication is authentic. In the case of a single message, such as a warning or alarm signal, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an ongoing interaction. such as the connection of a terminal to a host, two aspects are involved. First, at the tune of connection initiation, the service assures that the two entities are authentic, that is that each is the entity that it claims to be. Second, the service must assure that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties for the purposes of unauthorized transmission or reception. Two specific authentication services are defined in X.800:

- **Peer entity authentication:** Provides for the corroboration of the identity of a peer entity in an association. It is provided for use at the establishment of or at times during the data transfer phase of a connection. It attempts to provide confidence that an entity is not performing either a masquerade or an unauthorized replay of a previous connection.
- **Data origin authentication:** Provides for the corroboration of the source of a data unit. It does not provide protection against the duplication or modification of data units. This type of service supports applications like electronic mail where there are no prior interactions between the communicating entities.

2. Access Control

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

3. Data Confidentiality

Confidentiality is the protection of transmitted data from passive attacks With respect to the content of a data transmission, several levels of protection can be identified. The broadest service protects all user data transmitted between two users over a period of time. For example, when a TCP connection is set up between two systems this broad protection prevents the release of any user data transmitted over the TCP connection. Narrower forms of this service can also be defined, including the protection of a single message or even specific fields within a message. These refinements are less useful than the broad approach and may even be more complex and expensive to implement.

The other aspect of confidentiality is the protection of traffic flow from analysis. This requires that an attacker not be able to observe the source and destination, frequency, length, or other characteristics of the traffic on a communications facility.

4. Data Integrity

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

A connection-oriented integrity service, one that deals with a stream of messages, assures

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

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

5. Nonrepudiation

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

6. Availability Service

Both X.800 and RFC 2828 define availability to be the property of a system or a system resource being accessible and usable upon demand by an authorized system entity, according to performance specifications for the system (i.e., a system is avail-able if it provides services according to the system design whenever users request them). A variety of attacks can result in the loss of or reduction in availability. Some of these attacks are amenable to automated countermeasures, such as authentication and encryption, whereas others require some sort of physical action to prevent or recover from loss of availability of elements of a distributed system.

X.800 treats availability as a property to be associated with various security services. However, it makes sense to call out specifically an availability service. An availability service is one that protects a system to ensure its availability. This service addresses the security concerns raised by denial-of-service attacks. It depends on proper management and control of system resources and thus depends on access control service and other security services.

1.2.2 Types of Attacks

Q4. Explain different types of security attacks.

(OR)

Classify different types of attacks.

Ans : **(Imp.)**

I) Passive Attacks

Passive attacks are in the nature of eavesdropping on, or monitoring of, trans-missions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are release of message contents and traffic analysis,

(i) Release of message

The release of message contents is easily understood. 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.

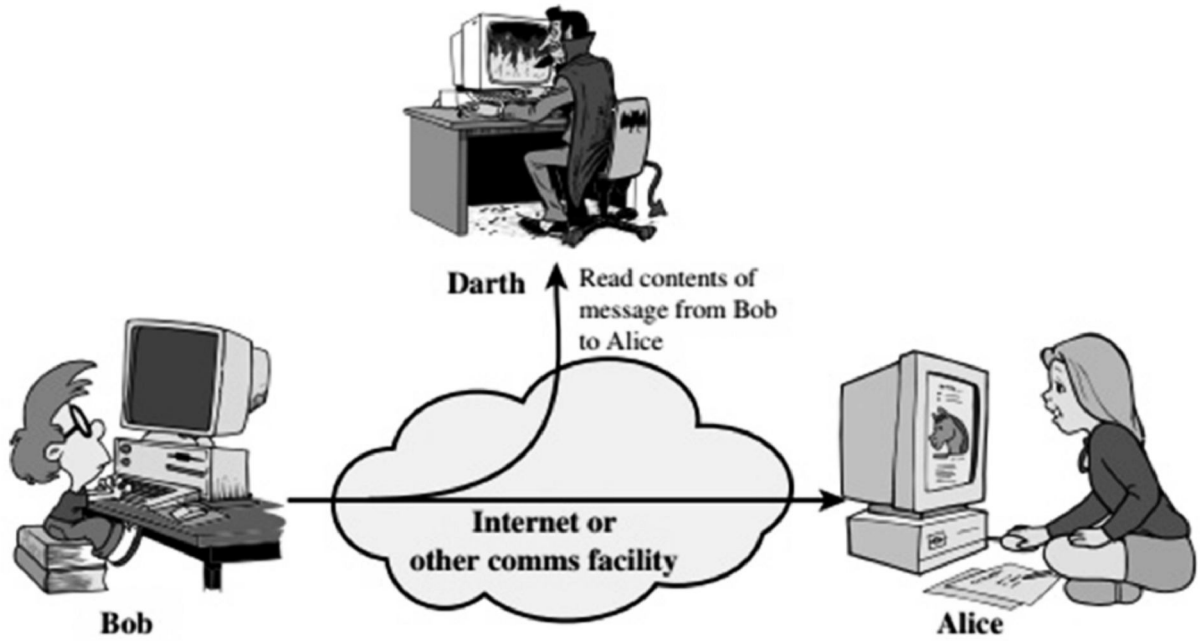


Fig.: Release of message contents

(ii) Traffic analysis

A second type of passive attack, traffic analysis, is subtler. 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.

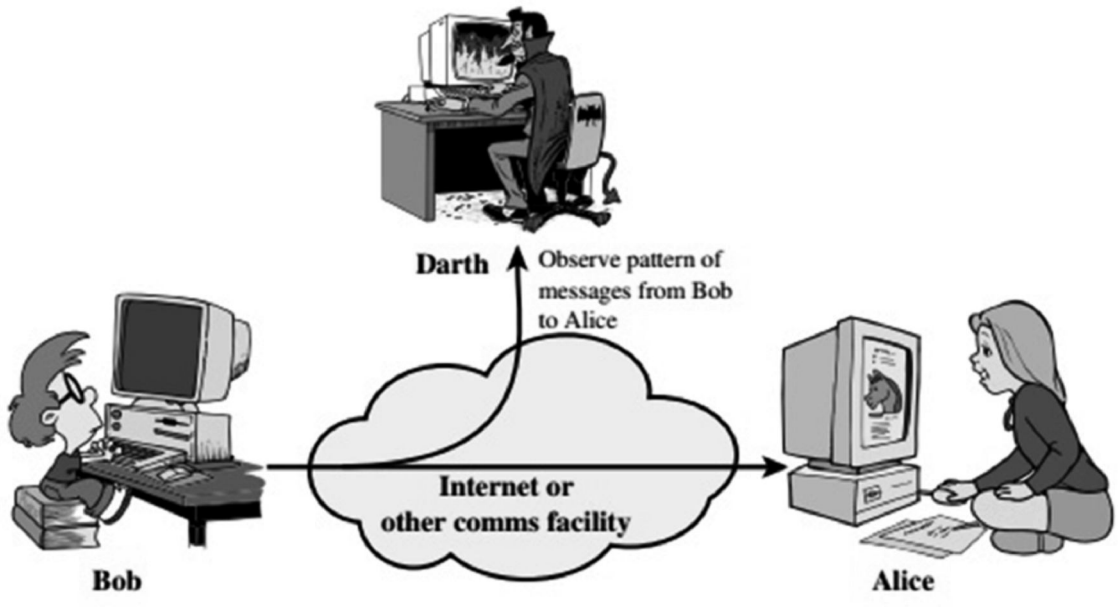


Fig.: Traffic analysis

II) 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:

(i) Masquerade

(ii) Replay

(iii) Modification of messages

(iv) Denial of service.

(i) Masquerade

A masquerade takes place when one entity pretends to be a different entity fig. 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.

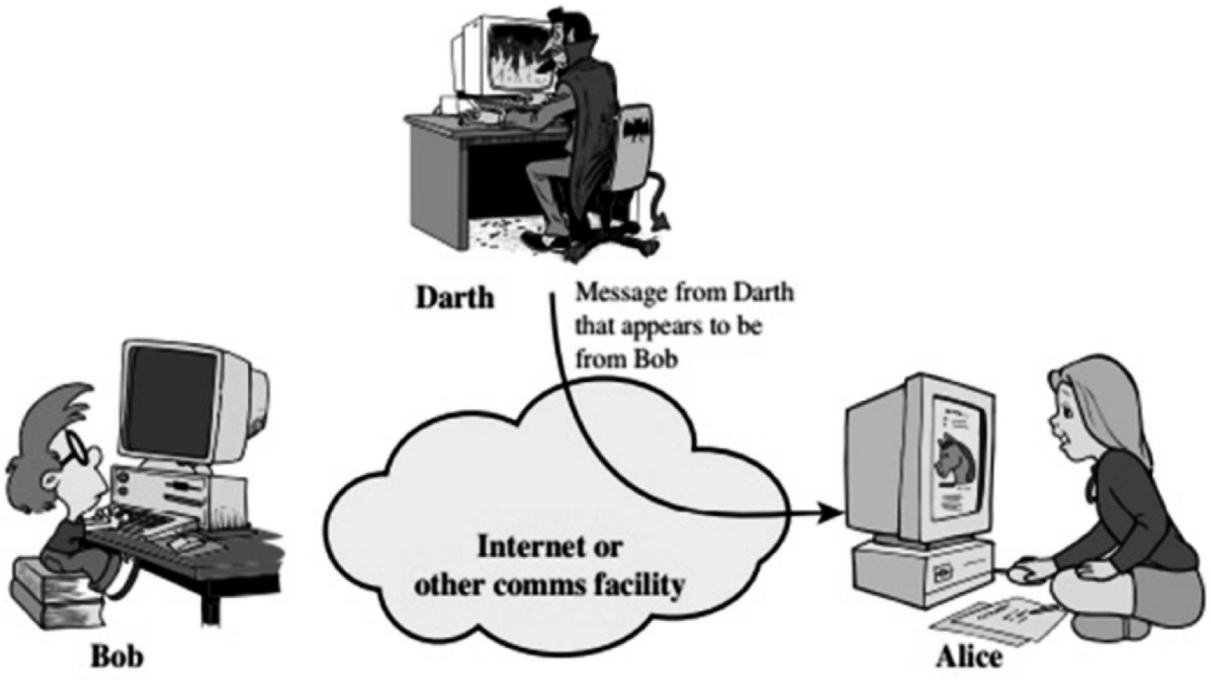


Fig.: Masquerade

(ii) Replay

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

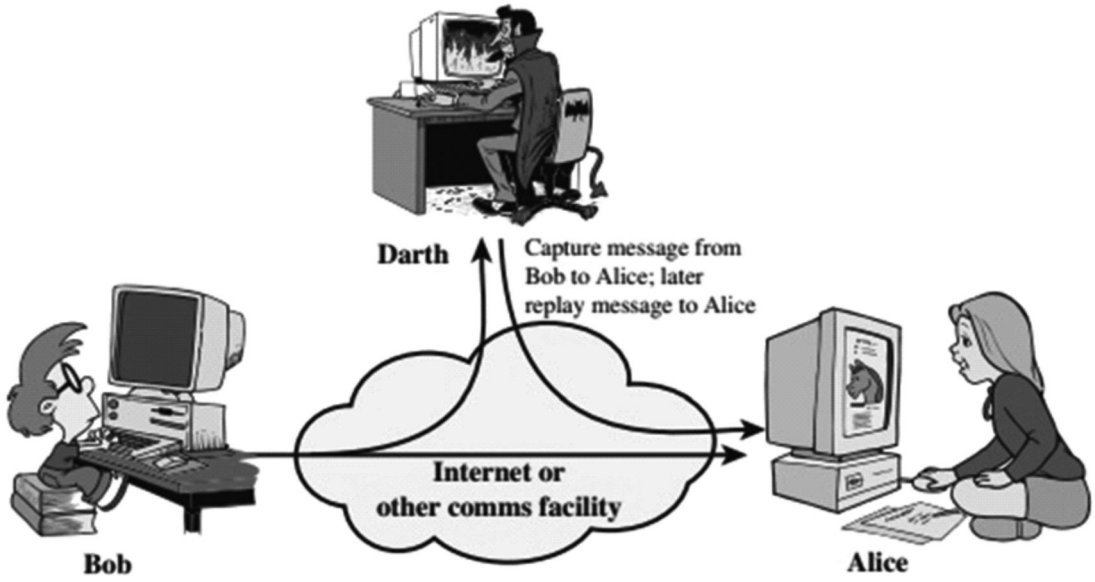
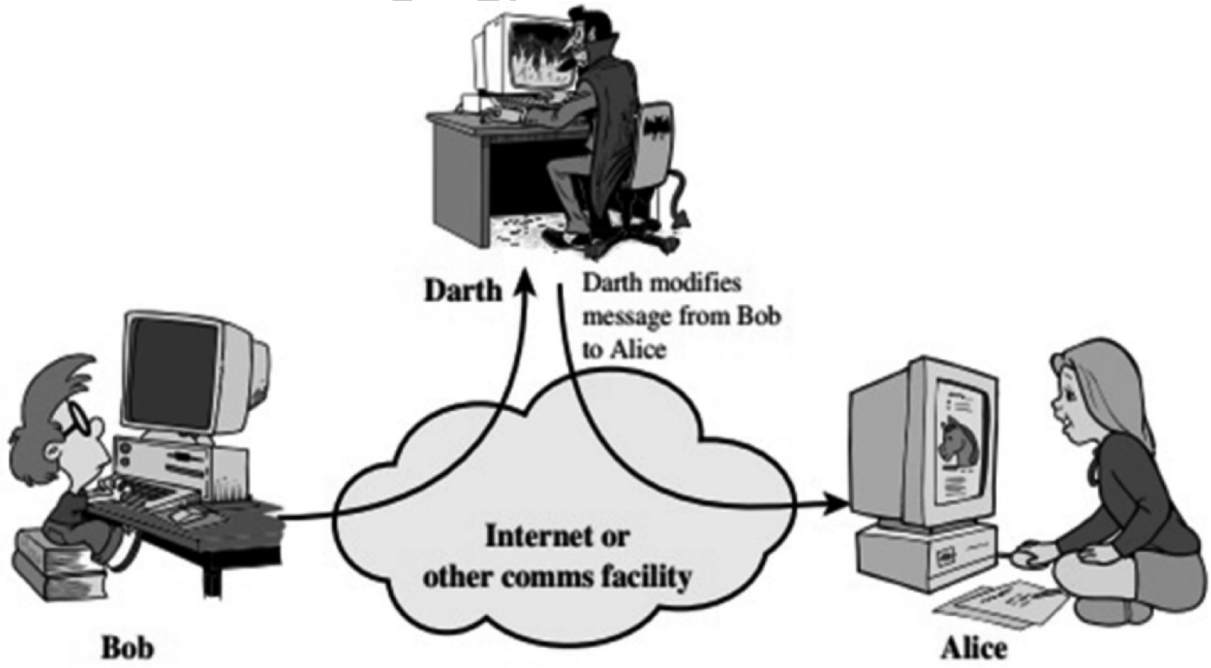


Fig.: Replay

(iii) Modification of messages

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 fig. 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.



Modification of messages

(iv) Denial of service

The denial of service prevents or inhibits the normal use or management of communications facilities fig. 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.

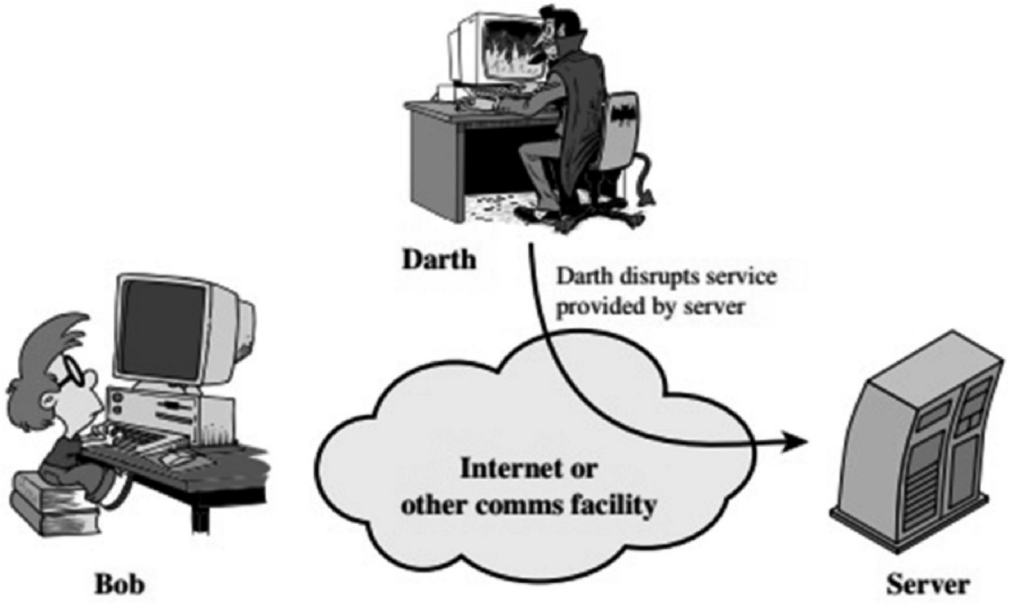


Fig.: Denial of service

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.

Q5. Describe the model for network security.

(OR)

Explain the model for network security.

Ans : **(Imp.)**

A model for much of what we will be discussing is captured, in very general terms, in fig. A message is to be transferred from one party to another across some sort of internet. The two parties, who are the principals in this transaction, must cooperate for the exchange to take place. A logical information channel is established by defining a route through the internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.

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

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

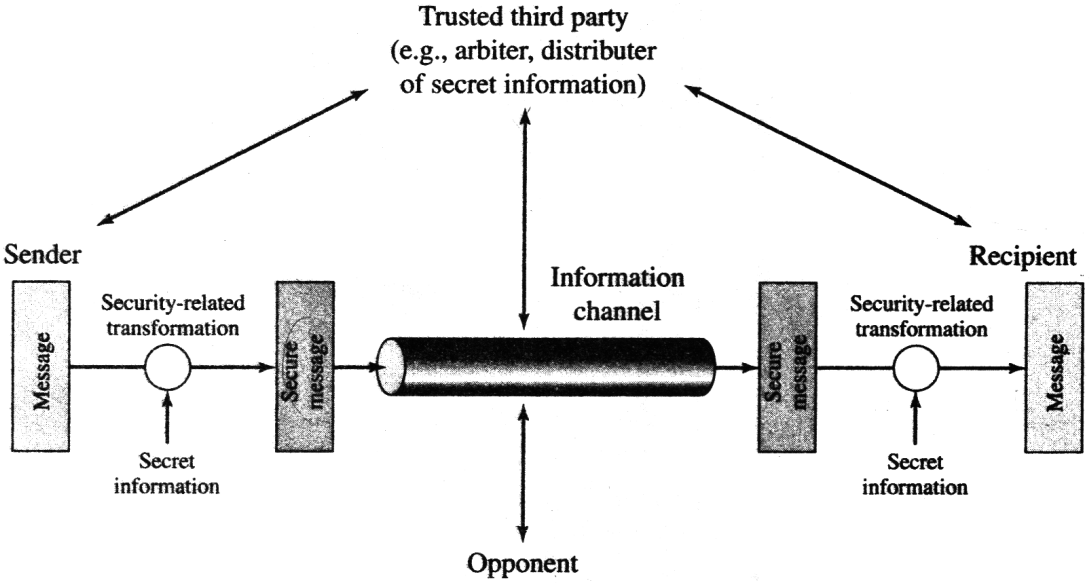


Fig.: Model for Network Security

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 algonthm 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.

Parts One through Three of this book concentrates on the types of security mechanisms and services that fit into the model shown in fig. However, there are other security-related situations of interest that do not neatly fit this model but that are considered in this book. A general model of these other situations is illustrated by Figure 1.6, which reflects a concern for protecting an information system from unwanted access. Most readers are familiar with the concerns caused by the existence of hackers, who attempt to penetrate systems that can be accessed over a network. The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system. Or, the intruder can be a disgruntled employee who wishes to do damage, or a criminal who seeks to exploit computer assets for financial gain (e.g., obtaining credit card numbers or performing illegal money transfers).

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 pre-sent 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.

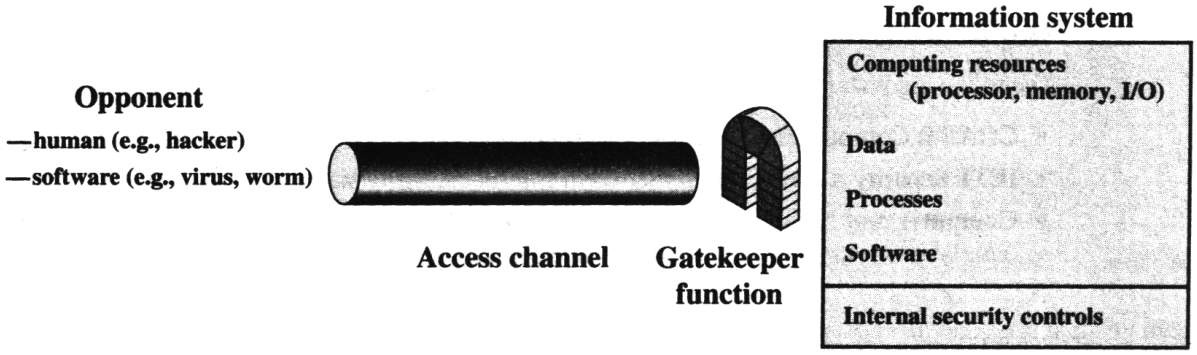


Fig.: Network Access Security Model

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

The security mechanisms needed to cope with unwanted access fall into two broad categories (see-fig. The first category might be termed a gatekeeper function. It includes password-based login procedures that are designed to deny access to all but authorized users and screening logic that is designed to detect and reject worms, viruses, and other similar attacks. Once either an unwanted user or unwanted software gains access, the second line of defense consists of a variety of internal controls that monitor activity and analyze stored information in an attempt to detect the presence of unwanted intruders.

1.3 DoS

Q6. What is Denial of service attack?

(OR)

Explain the concept of DoS.

Ans : **(Imp.)**

A Denial-of-Service (DoS) attack is an attack meant to shut down a machine or network, making it inaccessible to its intended users. DoS attacks accomplish this by flooding the target with traffic, or sending it information that triggers a crash. In both instances, the DoS attack deprives legitimate users (i.e. employees, members, or account holders) of the service or resource they expected.

Victims of DoS attacks often target web servers of high-profile organizations such as banking, commerce, and media companies, or government and trade organizations. Though DoS attacks do not typically result in the theft or loss of significant information or other assets, they can cost the victim a great deal of time and money to handle.

There are two general methods of DoS attacks flooding services or crashing services. Flood attacks occur when the system receives too much traffic for the server to buffer, causing them to slow down and eventually stop. Popular flood attacks include.

- **Buffer overflow attacks:** the most common DoS attack. The concept is to send more traffic to a network address than the programmers have built the system to handle. It includes the attacks listed below, in addition to others that are designed to exploit bugs specific to certain applications or networks
- **ICMP flood:** leverages misconfigured network devices by sending spoofed packets that ping every computer on the targeted network, instead of just one specific machine. The network is then triggered to amplify the traffic. This attack is also known as the smurf attack or ping of death.
- **SYN flood:** sends a request to connect to a server, but never completes the handshake. Continues until all open ports are saturated with requests and none are available for legitimate users to connect to.

Other DoS attacks simply exploit vulnerabilities that cause the target system or service to crash. In these attacks, input is sent that takes advantage of bugs in the target that subsequently crash or severely destabilize the system, so that it can't be accessed or used.

An additional type of DoS attack is the Distributed Denial of Service (DDoS) attack. A DDoS attack occurs when multiple systems orchestrate a synchronized DoS attack to a single target. The essential difference is that instead of being attacked from one location, the target is attacked from many locations at once. The distribution of hosts that defines a DDoS provide the attacker multiple advantages:

- He can leverage the greater volume of machine to execute a seriously disruptive attack
- The location of the attack is difficult to detect due to the random distribution of attacking systems (often worldwide)
- It is more difficult to shut down multiple machines than one
- The true attacking party is very difficult to identify, as they are disguised behind many (mostly compromised) systems

Modern security technologies have developed mechanisms to defend against most forms of DoS attacks, but due to the unique characteristics of DDoS, it is still regarded as an elevated threat and is of higher concern to organizations that fear being targeted by such an attack.

1.4 IP SPOOFING

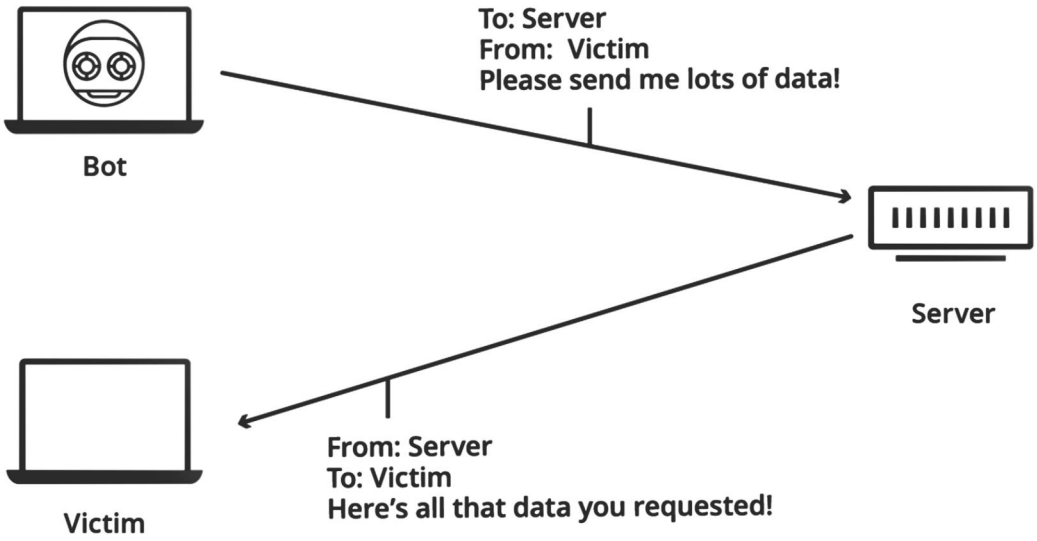
Q7. What is IP Spoofing?

Ans :

Meaning

IP spoofing is the creation of Internet Protocol (IP) packets which have a modified source address in order to either hide the identity of the sender, to impersonate another computer system, or both. It is a technique often used by bad actors to invoke DDoS attacks against a target device or the surrounding infrastructure.

Sending and receiving IP packets is a primary way in which networked computers and other devices communicate, and constitutes the basis of the modern internet. All IP packets contain a header which precedes the body of the packet and contains important routing information, including the source address. In a normal packet, the source IP address is the address of the sender of the packet. If the packet has been spoofed, the source address will be forged.



IP Spoofing is analogous to an attacker sending a package to someone with the wrong return address listed. If the person receiving the package wants to stop the sender from sending packages, blocking all packages from the bogus address will do little good, as the return address is easily changed. Relatedly, if the receiver wants to respond to the return address, their response package will go somewhere other than to the real sender. The ability to spoof the addresses of packets is a core vulnerability exploited by many DDoS attacks.

DDoS attacks will often utilize spoofing with a goal of overwhelming a target with traffic while masking the identity of the malicious source, preventing mitigation efforts. If the source IP address is falsified and continuously randomized, blocking malicious requests becomes difficult. IP spoofing also makes it tough for law enforcement and cyber security teams to track down the perpetrator of the attack.

Spoofing is also used to masquerade as another device so that responses are sent to that targeted device instead. Volumetric attacks such as NTP Amplification and DNS amplification make use of this vulnerability. The ability to modify the source IP is inherent to the design of TCP/IP, making it an ongoing security concern.

1.5 REPLAY

Q8. What is replay attack?

Ans :

Meaning

Replay Attack is a type of security attack to the data sent over a network.

In this attack, the hacker or any person with unauthorized access, captures the traffic and sends communication to its original destination, acting as the original sender. The receiver feels that it is an authenticated message but it is actually the message sent by the attacker. The main feature of the Replay Attack is that the client would receive the message twice, hence the name, Replay Attack.

Prevention from Replay Attack :

1. Timestamp method

Prevention from such attackers is possible, if timestamp is used along with the data. Supposedly, the timestamp on a data is more than a certain limit, it can be discarded, and sender can be asked to send the data again.

2. Session key method

Another way of prevention, is by using session key. This key can be used only once (by sender and receiver) per transaction, and cannot be reused.

1.6 Man-in-the-Middle Attacks

Q9. Explain the concept of Man-in-the-Middle attacks.

Ans : **(Imp.)**

Meaning

A man-in-the-middle attack is a type of eavesdropping attack, where attackers interrupt an existing conversation or data transfer. After inserting themselves in the "middle" of the transfer, the attackers pretend to be both legitimate participants. This enables an attacker to intercept information and data from either party while also sending malicious links or other information to both legitimate participants in a way that might not be detected until it is too late.

You can think of this type of attack as similar to the game of telephone where one person's words are carried along from participant to participant until it has changed by the time it reaches the final person. In a man-in-the-middle attack, the middle participant manipulates the conversation unknown to either of the two legitimate participants, acting to retrieve confidential information and otherwise cause damage.

Common abbreviations for a man-in-the-middle attack including MITM, MitM, MiM, and MIM.

Types

1. Email Hijacking

Attackers gain access to a user's email account and watch transactions to and from the account. When the time is right, for instance the user is exchanging funds with another party, the attacker takes advantage of the situation by attempting to intercept the funds by spoofing one or all members of the conversation.

2. Wi-Fi Eavesdropping

A passive way to deploy MITM attacks, Wi-Fi eavesdropping involves cyber hackers setting up public Wi-Fi connections, typically with an unsuspecting name, and gain access to their victims as soon as they connect to the malicious Wi-Fi.

3. Session Hijacking

session hijacking is when an attacker gains access to an online session via a stolen session key or stolen browser cookies.

4. DNS Spoofing

an attacker engages in DNS spoofing by altering a website's address record within a DNS (domain name server) server. A victim unknowingly visits the fake site and the attacker will attempt to steal their information.

5. IP Spoofing

similar to DNS spoofing, IP Spoofing sees an attacker attempt to divert traffic to a fraudulent website with malicious intent. Instead of spoofing the website's address record, the attacker disguises an IP (internet protocol) address.

1.7 General Threats to Computer Network

1.7.1 Worms, Viruses, -Trojans

Q10. Explain General Threats to Computer Network.

Ans : **(Imp.)**

Information Security threats can be many like Software attacks, theft of intellectual property,

identity theft, theft of equipment or information, sabotage, and information extortion.

Threat can be anything that can take advantage of a vulnerability to breach security and negatively alter, erase, harm object or objects of interest.

Software attacks means attack by Viruses, Worms, Trojan Horses etc. Many users believe that malware, virus, worms, bots are all same things. But they are not same, only similarity is that they all are malicious software that behaves differently.

Malware is a combination of 2 terms- Malicious and Software. So Malware basically means malicious software that can be an intrusive program code or anything that is designed to perform malicious operations on system. Malware can be divided in 2 categories:

I. Infection Methods

II. Malware Actions

I) Infection Methods

Malware on the basis of Infection Method are following:

1. Virus

They have the ability to replicate themselves by hooking them to the program on the host computer like songs, videos etc and then they travel all over the Internet. The Creeper Virus was first detected on ARPANET. Examples include File Virus, Macro Virus, Boot Sector Virus, Stealth Virus etc.

2. Worms

Worms are also self-replicating in nature but they don't hook themselves to the program on host computer. Biggest difference between virus and worms is that worms are network-aware. They can easily travel from one computer to another if network is available and on the target machine they will not do much harm, they will, for example, consume hard disk space thus slowing down the computer.

3. Trojan

The Concept of Trojan is completely different from the viruses and worms. The name Trojan is derived from the 'Trojan Horse' tale in Greek mythology, which explains how the Greeks were able to enter the fortified city of Troy by hiding their soldiers in a big wooden horse given to the Trojans as a gift. The Trojans were very fond of horses and trusted the gift blindly. In the night, the soldiers emerged and attacked the city from the inside.

Their purpose is to conceal themselves inside the software that seem legitimate and when that software is executed they will do their task of either stealing information or any other purpose for which they are designed.

They often provide backdoor gateway for malicious programs or malevolent users to enter your system and steal your valuable data without your knowledge and permission. Examples include FTP Trojans, Proxy Trojans, Remote Access Trojans etc.

4. Bots

Can be seen as advanced form of worms. They are automated processes that are designed to interact over the internet without the need for human interaction. They can be good or bad. Malicious bot can infect one host and after infecting will create connection to the central server which will provide commands to all infected hosts attached to that network called Botnet.

II) Malware Actions

Malware on the basis of Actions:

1. Adware

Adware is not exactly malicious but they do breach privacy of the users. They display ads on a computer's desktop or inside individual programs. They come attached with free-to-use software, thus main source of revenue for such developers. They monitor your interests and display relevant ads. An attacker can embed malicious code inside the software and adware can monitor your system activities and can even compromise your machine.

2. Spyware

It is a program or we can say software that monitors your activities on computer and reveal collected information to an interested party. Spyware are generally dropped by Trojans, viruses or worms. Once dropped they install themselves and sits silently to avoid detection.

One of the most common example of spyware is KEYLOGGER. The basic job of keylogger is to record user keystrokes with timestamp. Thus capturing interesting information like username, passwords, credit card details etc.

3. Ransomware

It is type of malware that will either encrypt your files or will lock your computer making it inaccessible either partially or wholly. Then a screen will be displayed asking for money i.e. ransom in exchange.

4. Scareware

It masquerades as a tool to help fix your system but when the software is executed it will infect your system or completely destroy it. The software will display a message to frighten you and force to take some action like pay them to fix your system.

5. Rootkits

are designed to gain root access or we can say administrative privileges in the user system. Once gained the root access, the exploiter can do anything from stealing private files to private data.

6. Zombies

They work similar to Spyware. Infection mechanism is same but they don't spy and steal information rather they wait for the command from hackers.

- **Theft of intellectual property:** means violation of intellectual property rights like copyrights, patents etc.
- **Identity theft:** means to act someone else to obtain person's personal information or to access vital information they have like accessing the computer or social media account of a person by login into the account by using their login credentials.
- **Theft of equipment and information:** is increasing these days due to the mobile nature of devices and increasing information capacity.
- **Sabotage:** means destroying company's website to cause loss of confidence on part of its customer.
- **Information extortion:** means theft of company's property or information to receive payment in exchange. For example ransomware may lock victims file making them inaccessible thus forcing victim to make payment in exchange. Only after payment victim's files will be unlocked.

These are the old generation attacks that continue these days also with advancement every year. Apart from these there are many other threats. Below is the brief description of these new generation threats.

- **Technology with weak security:** with the advancement in technology, with every passing day a new gadget is being released in the market. But very few are fully secured and follows Information Security principles. Since the market is very competitive Security factor is compromised to make device more up to date. This leads to theft of data/ information from the devices
- **Social media attacks:** in this cyber criminals identify and infect a cluster of websites that persons of a particular organization visit, to steal information.
- **Mobile Malware:** there is a saying when there is a connectivity to Internet there will be danger to Security. Same goes for Mobile phones where gaming applications are designed to lure customer to download the game and unintentionally they will install malware or virus on the device.

- **Outdated Security Software:** with new threats emerging everyday, updation in security software is a prerequisite to have a fully secured environment.
- **Corporate data on personal devices:** these days every organization follows a rule BYOD. BYOD means Bring your own device like Laptops, Tablets to the workplace. Clearly BYOD pose a serious threat to security of data but due to productivity issues organizations are arguing to adopt this.
- **Social Engineering:** is the art of manipulating people so that they give up their confidential information like bank account details, password etc. These criminals can trick you into giving your private and confidential information or they will gain your trust to get access to your computer to install a malicious software that will give them control of your computer. For example email or message from your friend, that was probably not sent by your friend. Criminal can access your friends device and then by accessing the contact list, he can send infected email and message to all contacts. Since the message/ email is from a known person recipient will definitely check the link or attachment in the message, thus unintentionally infecting the computer.

Q11. Describe the classification of virus.

(OR)

Explain the classification of virus.

Ans : **(Imp.)**

1. Boot Sector Virus

The basic boot sequence has two important steps after power on:

(i) Primary Boot

The ROM based instructions first run, and then after a self test identifies the boot device. The boot block is then read, and the control is transferred to the loaded code.

(ii) Secondary Boot

The code loaded during the primary boot loads a program that understands the boot device's file system structure. This is called the secondary boot and the operating system kernel is loaded during this step.

A boot sector virus, or Boot Sector Infector (BSI) is a virus that infects by copying itself into the master boot block. The content of the former boot block is copied else where in the disk. Thus, the vims after completing the infection and other tasks can complete the booting process.

The advantage of a boot sector vims is that it launches before any anti-vims starts or the operating system is loaded. However BSI's are now rare. Machines are not booted with boot floppies, and operating systems prevent writing to the disk's boot block.

Examples of boot vims were Michelangelo and Stone.

2. Executable File Infectors

Afile infector is a vims which infects files which the operating system considers to be executables. Binary executables are the commonest targets of these viruses. The vimses can be placed at the beginning, end, over-written or inserted into a file.

Viruses which places its codes at the beginning of a target executable code is often called Prepending vims. Older file formats like COM, MS-DOS when executed, the entire file gets loaded and the execution starts from the beginning of the loaded file. Thus the vims gets control first when the infected file is mn. An example of a COM prepending vims is the Hungarian virus Polimer.512 A, which is 512 bytes long and prepends itself at the beginning of the executable.

In contrast, there are some vimses which append itself at the end of the target file. This is even easier than the prepending vims. These vimses are called appending virus. Many executable files have a header which points to the start location of a file. The vims can change this start location to point to the vims code and then jump to the original start location when it has completed its work. An example of this vims is Vienna.

3. Overwriting Virus

Overwriting vimses, as the name suggests, overwrites the target files with their own copy. This avoids change in the size of the file size, which occurs in the prepending or the appending vims discussed above. This is advantageous for the vims as a change in size of a file may be used to identify a vims. However, blind overwriting can also be detected easily by the anti-vims. The vims can thus adopt several techniques:

- The virus can overwrite sections of repeated values in the program's data. After finishing its activity the vims can restore the repeated values.
- The virus can save the original content of the file which it is overwriting, similar to the boot sector vims. Viruses can take advantage of the fact that the operating system allocates more data than required for a file. This extra space can be used stealthily by the vims to evade detection.
- Due to alignment to the page boundary, parts of executable files are padded. The unused space can be used by the virus to be silently present in the target code.
- A virus can also apply compression methods to create space for itself. But enough space needs to be created for itself and also the decompression program. The decompression program is used by the virus after finishing its activity to restore the original code.

4. Companion Virus

Some types of viruses do not modify the infected code. A companion virus is one that installs itself such that it is executed before the target code. It takes advantage of the process by which the operating system or shell searches for executable files. One approach used in MS-DOS is to give the virus the same name as the target, but change the extension from .EXE to .COM. In MS-DOS if an executable named foo is searched, the files looked after consecutively are foo.com, foo.exe and foo.bat. This technique was used by the Globe virus around 1992. The victim trying to execute a file, generally types without the extension. In such a case the operating systems give priority to the COM extension over the .EXE extension, thus triggering the malicious program.

Yet another important class of functions is known as the Macro Virus, which takes advantage of the "macros" embedded in some editing applications, like word processors. Macros are short snippets of codes written in a language which is interpreted by the application. The language is equipped enough to write a virus much easily compared to that using a low level language. An example of such a virus was the Concept virus, which targeted the set of macros which are used by all edited documents in Microsoft Word. For example, the FileSaveAs macro applies to all documents which we edit in word. The effect of the macro virus was that it takes an uninfected virus and as the user uses the File Save As menu, the file gets infected as the macro virus gets triggered.

Next we consider classifications of virus based on the way the virus program conceals or hides itself from an anti-virus program. The types of virus classified based on the concealment strategy can be Encryption, Oligomorphism, Polymorphism and Metamorphism. We briefly describe these types one by one.

5. Encryption

The virus body if not encrypted, is trivial to detect and analyze. However, if the virus body, consisting of the infection, trigger and payload, is encrypted then it becomes difficult to be detected. To execute the virus, the encrypted body thus needs to be decrypted. Hence encrypted virus has a dedicated decryptor loop, which decrypts the virus body and transfer the control to it. This loop is typically much smaller in size compared to the body, and thus provides a smaller profile to the antivirus.

For encryption of the body generally the following methods are adopted. In general, the encryptions used are more of obfuscation techniques rather than strong cryptographic mechanisms.

(i) Simple Transformations

The transformations in the virus body could be a sequence of simple logical and arithmetical operations. They could include logical not, arithmetic shift, rotations, incrementations and decrementations. Note that these operations are key-less transformations.

(ii) Key Mixing

A static or variable key is mixed with the body using some reversible transformations, like xoring or addition. The key could either be a static key for the entire body or it could vary along with the decryption. The body could be divided into some components, and the key for the / component could be derived from the key of the previous component by adding with the content of the I^{th} component of the body.

(iii) Substitution Cipher

Look up tables could also be used to achieve the encryption. The component of the virus body could be looked into fixed tables and transformed accordingly.

(iv) Strong Encryption

With the development of cryptographic libraries like openssl, virus bodies can also decrypt using these function calls without much overhead.

The problem with these encryption schemes is that the encrypted output remains the same. This increases the possibility of being detected by anti-virus routines. Hence, a natural way out is to use varying keys, but then the decryptor loop needs to be updated with the changed key.

6. Oligomorphism

When the virus uses a varying key, the virus body gets changed with every encryption. However, the decryptor loop does not change for most of the part. This again gives an avenue to the anti-virus routines to spot the virus.

A scheme which is used by virus programs is that instead of one decryptor loop, a pool of decryptor loops are used. The virus selects one of these decryptor loops. Thus, the entire virus changes and hence become harder to detect. But the increase in difficulty is only marginal as the detecting software now needs to check for the variants of the decryptor loop. For example, the virus Memorial has 96 decryptors.

7. Polymorphism

A polymorphic virus is almost the same as the oligomorphic virus. But in these viruses, the total number of decryptor loops are extremely large. For example, a virus called Tremor has almost six billion possible decryptor loops.

The virus changes its decryptor loop by using a mutation engine, which changes the decryptor loop every encryption. The mutation engine randomly (or pseudo-randomly) selects a trick from a repository of possible transformations to change the loop.

Various methods used for code obfuscation and by compilers for optimizing codes. Some of the methods which are used for writing virus are Instruction equivalence, Instruction sequence equivalence, Instruction reordering, register renaming, concurrency, writing convoluted programs, inlining and outlining of function calls. Next we briefly explain these techniques:

(i) Instruction Equivalence

This method takes advantage of the fact that often many instructions are provided which can be used to the same effect. Like the

instructions, clear rl (clear the register rl) is the same as xor rl rl (perform a xor between the contents stored in register rl with itself and store the result, which is zero, back to register rl.

(ii) Instruction Sequence Equivalence

While in Instruction equivalence, two instructions are equivalent, in Instruction sequence equivalence, it is the sequence of instructions which are the same. Thus an assignment x = 1, can be stated equivalently as the sequence of instructions: x= l0, x = x – 9. Note that this technique can be adopted both for assembly level programming as well as for high level languages.

(iii) Instruction Reordering

A sequence of instructions may have dependencies among themselves. Using the independence among some instructions, they can be performed in different orders. These techniques are well studied for instruction level parallelism. Register renaming is a minor method which indicates change of the names of the registers that are used in the instructions. It is an extremely simple technique but may confuse anti virus routines.

(iv) Concurrency

The original code can be separated into multiple threads of execution. This method can greatly complicate automatic analysis by the anti virus. For example, a sequence of operations, a = funcl(), b = func2( ) and c = a + b can be implemented using two threads. One of the thread computes a, while the other evaluates b. Finally, the sum c is determined. Hence we have,

```
start thread T
a=funcl( ); wait for signal
c=a + b;
T:
b=func2( );
send signal;
exit thread T;
```

This procedure naturally takes advantage of the scope of concurrency in the program.

Inlining and Outlining of Functions Code inlining is the procedure to replace subroutine calls with subroutine codes. It is normally used to avoid function call overheads. Outlining is the reverse operations. Writing convoluted codes or spaghetti codes is also a technique used for varying the decryptor loop body of the virus.

8. Metamorphism

These viruses do not use decryption but changes the entire body of the virus using the techniques used by polymorphic viruses for their decryptor loop. Thus, they are often described as “Metamorphics

are body polymorphics". Thus the mutation engine of a metamorphic virus needs to change from infection to infection.

Win 32/Apparition virus carries its source and drops it whenever it can find a compiler installed on the machine. This is a real threat for Linux based systems, which has C compilers placed in a typical installation of the operating system. The virus inserts junk code into and removes itself from the source, and then recompiles itself. Thus, each new generation of the virus is completely different and thus makes detection a difficult problem.

In December 1998, Vecna, a noted (notorious?) Virus writer, created the Win95/Regswap virus. Regswap achieved metamorphosis via register usage exchange. Any part of the virus body will use different registers but the same code.

For example consider the following code:

```
5A              pop     edx
BF04000000      mov     edi,    0004h
8BF5            mov     esi,    ebp
B80C000000      mov     eax,    000ch
81C288000000    add     edx,    008h
8B1A            mov     ebx,    [edx]
899C8618110000  mov [esi+eax*4+00001118], ebx
```

A new generation of the above code with some register exchanges and renaming are as follows:

```
58              pop     eax
BB04000000      mov     ebx,    0004h
8BD5            mov     edx,    ebp
BF0C000000      mov     edi,    000ch
81C088000000    add     eax,    008h
8B30            mov     esi,    [edx]
89B4BA18110000  mov [esi+eax*4+00001118], esi
```

The similarity in the code generation may be observed from the above two generations. The similarities may be used for detection of the virus. This may involve string matching techniques, to obtain the wild string which will help to identify the virus body.

More recent viruses like Win95/Zperm family, which appears in 2000, inserts random jump instructions into a code. The jmps will be inserted to point to the next instruction of the virus. The mutation of the virus involves removal and addition of jump and garbage instructions. A code with n instructions can be rearranged in at least n! (read n factorial) ways, with the inserted jmp instructions helping in the reordering of the instructions. Most polymorphic viruses decrypt themselves to a single constant body in the memory. However, metamorphic viruses do not. Thus the detection of such viruses needs to be algorithmic, even in the memory. String matching methods do not work as the virus bodv is nowhere constant.

In case of polymorphic viruses, there is a snap shot of the complete decrypted virus body. Anti-virus programs use generic decryption engines based on code emulation to observe this snap-shot. However, in case of metamorphic viruses this particular moment does not exist.

The development of metamorphic viruses has given developers of viruses lots of power. Development of suitable anti-virus techniques is an increasingly growing challenge.

Q12. Explain the concept of worms.

(OR)

Describe briefly about worms.

Ans : **(Imp.)**

The worm programs exposed the security flaws of the standard functions provided by UNIX. We shall discuss some problems specific to some UNIX utilities. A family of routines in the standard C library takes inputs to buffers without checking for bounds. Examples include gets, scanf, fscanf, sscanf and other such routines. They may thus be used to overrun buffers unless the user explicitly takes care of the number of characters. Careless usage of the routines like sprintf and usage of strcat and strcpy instead of strncat and strncpy may also overflow buffers. The problem with these codes is they are functionally correct and seemingly harmless. However, the usage of such codes in networking or trusted programs can cause exploits to work by carefully doctoring inputs.

In order to circumvent this issue patches were developed. The first measure adopted was to replace all such codes by bounded versions. These versions accept values for bounds on their buffer arguments. Next, all servers and trusted applications should be checked for usage of these bounded versions of codes. For example, consider the fingerd program which runs as a daemon to service remote requests using the finger protocol, an utility which allows users to obtain information about other users. The fingerd servers used the gets call, and thus patches had to be developed for these programs. These revised versions of programs do not make use of the original gets commands and were devoid of function calls which fill buffers without checking for bounds.

The other way of running a worm is by targeting the sendmail program. The sendmail program is used to route mail in a heterogeneous network. When the sendmail program operates as a daemon process, the program “listens” on a TCP port 25 in order to deliver mails using Simple Message Transfer Protocol (SMTP). The worm worked by exploiting the debugging option in the program. The worm would specify the debug command to sendmail and then specify a set of commands instead of a user

address as the recipient of a message. Normally, this is not allowed but it is kept to allow easy testing of the complicated sendmail program. New versions of this program have been released where the debug option has been disabled. Or, it requests the user to enter the root password in order to avail of the debug option.

The worms also attempt to determine the passwords which the users provide. The Unix passwords previously were stored as plaintext in a "password file". The system protected the passwords by controlling the access, such that the passwords were visible only to system administrators and privileged users. However, due to unintentional errors or wrong programming, this practice can be dangerous. There is an instance which occurred in the early 1960s, when two system administrators at MIT were editing the password file and the daily message that gets printed on every ones terminal during login. However due to a software error, both the files got swapped and for a time the password file was printed on every terminal when it was logged in.

In order to avoid such problems, UNIX does not keep actual passwords anywhere on the system Instead, UNIX uses a program called crypfi) to encrypt a block of 0 bits with the user password. The result of the encryption is stored in the /etc/passwd file. When a user tries to login, the password is not decrypted (it was actually never encrypted). Instead the login program takes a block of zeros anc compares the newly transformed block with the block stored in /etc/passwd file. If there is a match the user is allowed in by the system.

The algorithm that crypt( ) uses is based on the Data Encryption Standard (DES), developed by National Institute of Standards and Technology (NIST). The crypt( ) functions takes the user password as the encryption key and uses it to encrypt a 64 bit block of 0's. The resulting 64 bit ciphertext is then encrypted again with the password, the process being repeated 25 times. The final 64 bits are unpacked into a string of 11 printable characters that are stored in the /etc/passwdfile. The ability of DES function to resist against cryptanalysis, has lead to the fact that the only way of breaking the UNIX password security is to do a brute force search or a dictionary attack.

A dictionary attack is launched by the adversary by recording possible passwords chosen from say English and numbers in a file. Then they are encrypted by the UNIX crypt function and verified with the /etc/passwd file for any matches.

At the time of design of DES, the security of 56 bits brute force search was considered to be high However, in today's world with the progress of VLSI designs and parallel processors systems have been developed which can quite efficiently do such a search. The recent DES cracker named COPACOBANA was built in 2006 by teams from Universities of Bochum and Kiel. COPACOB ANA consists of 120 Field programmable gate arrays (FPGAs) of type XILINX Spartan3-1000 running in parallel. The cost of the machine is approximately $10,000, which is a reduction by a factor of 25 over previous such machines Since 2007, SciEngines GmbH, a spin-off company of the two project partners of COPACOBANA has enhanced and developed successors of COPACOBANA. In 2008, their COPACOBANA R1VYERA reduced the time to break DES to less than one day, using 128 Spartan-3 5000's.

In order to reduce such attacks, an approach was taken long back, which was to shadow the password. The hashed passwords (they are not really encrypted) are stored in a file, which is visible to only the system administrator, and a privileged call performs encryptions and comparisons with an appropriate delay. Additionally, often a threshold is set, which exceeded an alarm is raised by the system.

In addition the password is often "salted" to ensure that the result of the DES encryption gets changed. The DES salt is a 12 bit number, between 0 and 4095, which slightly changes the result of the DES function. When the password is changed with the /bin/passwd program, a salt is selected depending on the time of the day. The salt is converted into a two-character string and is stored in the /etc/passwd file along with the encrypted password. The purpose of the salt is thus that the same password can be stored in 4096 ways in th d etc/passwd file and thus increase the complexity of a dictionary attack 4096 times.

Worms can be classified by the primary method they use for transport. They can be divided into the following types:

1. **IM worms:** Worms using instant messaging is called IM worms.
2. **email worms:** Worms using email as a means of spreading are called email worms.

UNIT II

Secret Key Cryptography

DES, Triple DES, AES, Key distribution, Attacks.

Public Key Cryptography

RSA, ECC, Key Exchange (Diffie-Hellman), Java Cryptography Extensions, Attacks.

2.1 Secret Key Cryptography

2.1.1 DES

Q1. Describe the concept of DES algorithm.

(OR)

Explain the concept of DES.

Ans : **(Imp.)**

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 to reverse the encryption.

DES Encryption

The overall scheme for DES encryption is illustrated in figure. As with any encryption scheme, there are two inputs to the encryption function: the plain text to be encrypted and the key. In this case, the plain text must be 64 bits in length and the key is 56 bits in length.

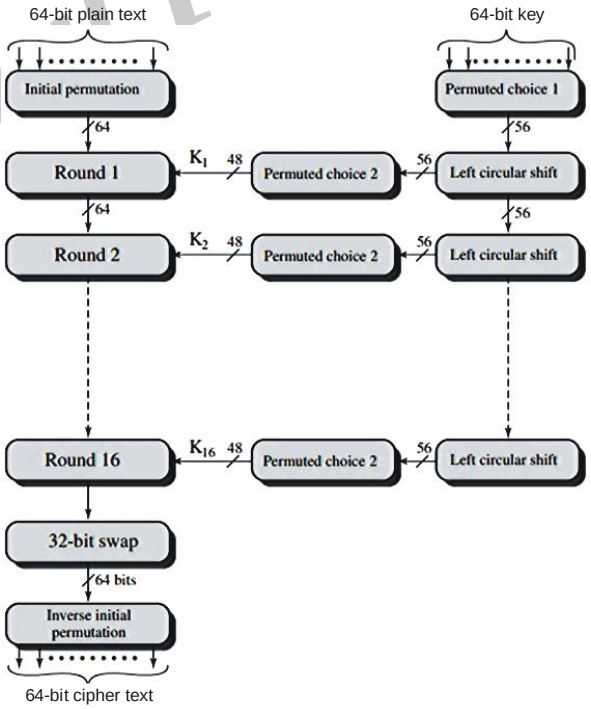


Fig. : General Depiction of DES Encryption Algorithm

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 and right halves of the output are swapped 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

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

where M_i is a binary digit. Then the permutation X = (IP(M) is as follows :

M_{58}	M_{50}	M_{42}	M_{34}	M_{26}	M_{18}	M_0	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

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

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

$$Li = Ri - 1$$

$$Li = Ri - 1$$

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

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.

The role of the S-boxes in the function F is illustrated in Figure. The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. These transformations are defined in Table, which is interpreted as follows: The first and last bits of the input to box Si form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for Si. The middle four bits select one of the sixteencoloumns. The decimal value in the cell selected by the row and column is then converted to its 4-bit representation 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 may be useful in understanding themapping. The figure shows the substitution for row 0 of box S1.

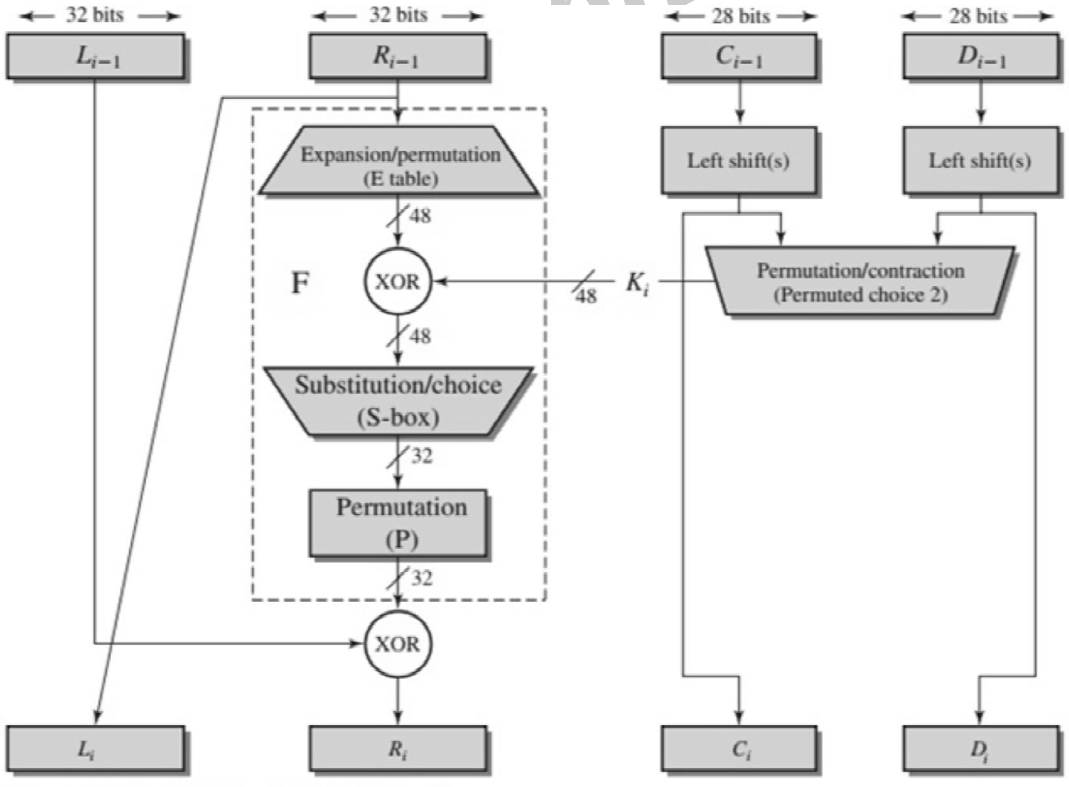


Fig. : Single Round of DES Algorithm

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 ...

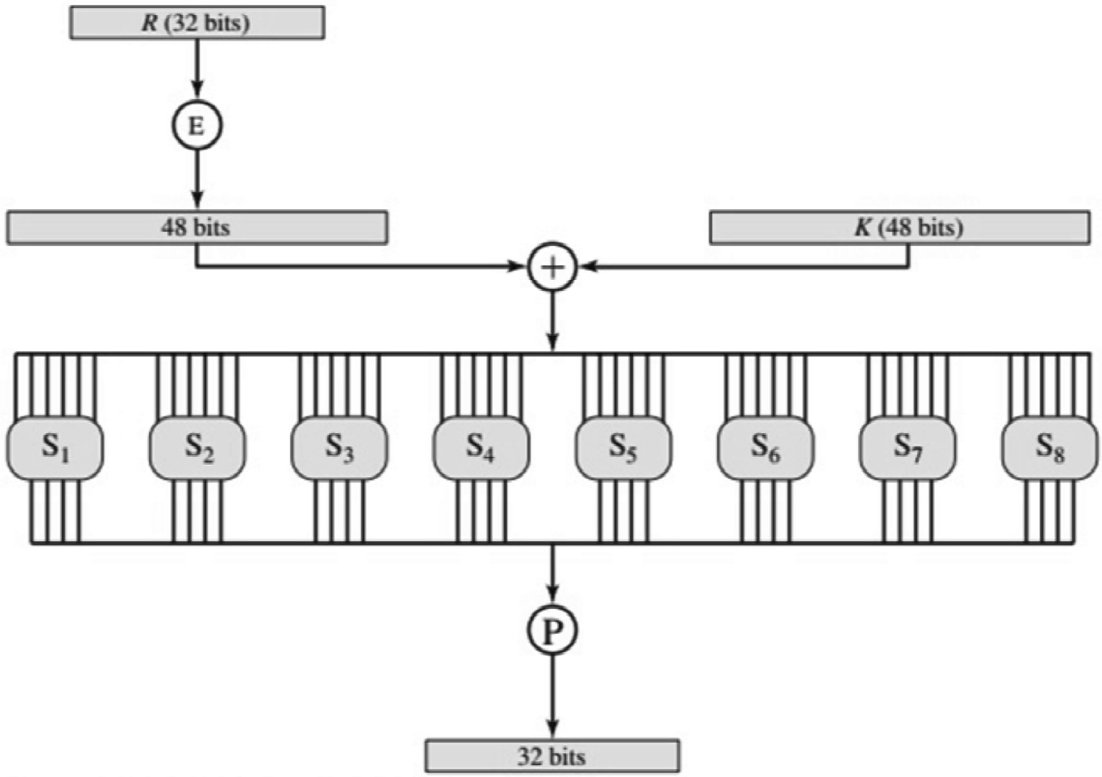


Fig. : Single Round of DES AlgorithmC

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 from the eight S-boxes is then permuted, so that on the next round, the output from each S-box immediately affects as many others as possible.

Key Generation

We see that a 64-bit key is used as input to the algorithm. The bits of the key are numbered from 1 through 64; every eighth bit is ignored, as indicated by the lack of shading in Table. The key is first subjected to a permutation governed by a table labelled Permuted Choice One. 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, as governed by Table. These shifted values serve as input to the next round. They also serve as input to the part labelled Permuted Choice Two, 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.

S_1

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
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 (PCI)

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

Q2. 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 as follows:

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 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 combined 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.

It 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 the two 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

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 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 few rounds.

Round		δ
	02468aceeca86420 02468aceeca86420	0
1	3cf03c0fbad22845 3cf03c0f9ad628c5	3
2	bad2284599e9b723 9ad628c59939136b	11
3	99e9b7230bae3b9e 9939136b768067b7	25
4	0bae3b9e42415649 768067b75a8807c5	29
5	4241564918b3fa41 5a8807c5488dbe94	26
6	18b3fa419616fe23 488dbe94aba7fe53	26
7	9616fe2367117cf2 aba7fe53177d21e4	27
8	67117cf2c11bfc09 177d21e4548f1de4	32

Round		δ
9	c11bfc09887fbc6c 548f1de471f64dfd	34
10	887fbc6c600f7e8b 71f64dfd4279876c	36
11	600f7e8bf596506e 4279876c399fdc0d	32
12	f596506e738538b8 399fdc0d6d208dbb	28
13	738538b8c6a62c4e 6d208dbbb9bdeeaa	33
14	c6a62c4e56b0bd75 b9bdeeaad2c3a56f	30
15	56b0bd7575e8fd8f d2c3a56f2765c1fb	33
16	75e8fd8f25896490 2765c1fb01263dc4	30
IP^{-1}	da02ce3a89ecac3b ee92b50606b62b0b	30

Table : Avalanche Effect in DES: Change in Key

Q3. Explain the strength of DES.

Ans : **(Imp.)**

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 a thousand 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.

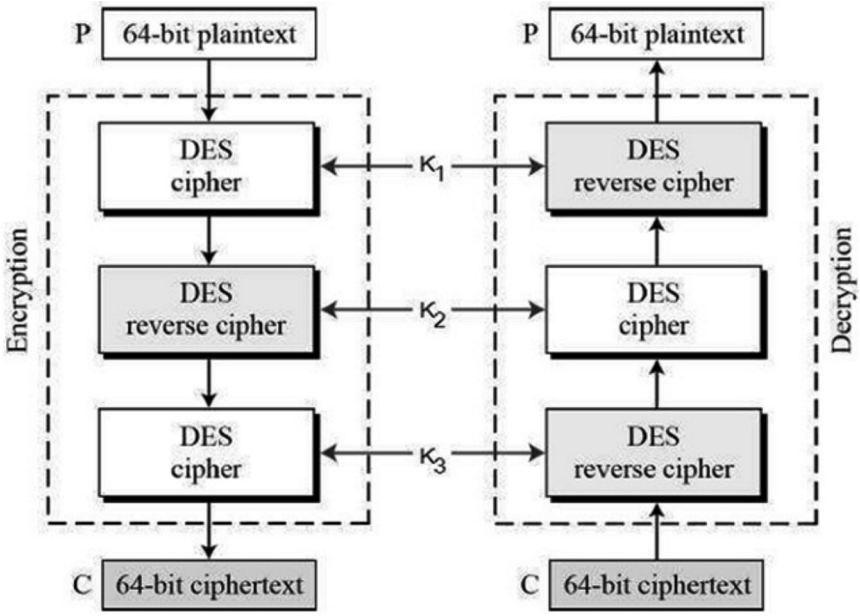
2.1.1.1 Triple DES

Q4. Explain triple DES with two keys.

Ans : **(Imp.)**

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.

2.1.2 AES

Q5. Define AES. Explain the features of AES.

Ans : **(Imp.)**

Meaning

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.

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.

Q6. Explain about the structure of AES.

Ans :

The AES cipher in more detail, indicating the sequence of transformations 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 of three 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 the key 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, an inverse 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.

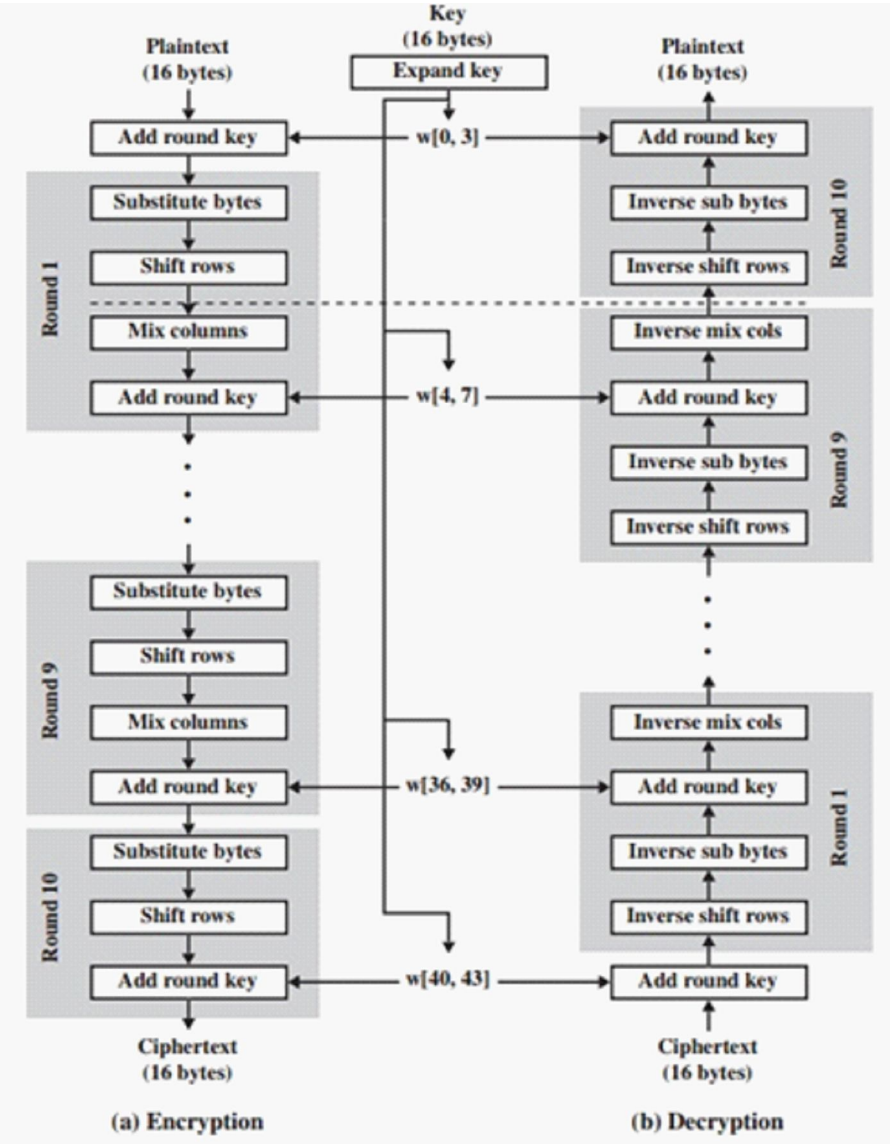


Figure : AES Encryption and Decryption

Q7. Explain AES Round function.

(OR)

Explain about AES encryption round key.

Ans : **(Imp.)**

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]

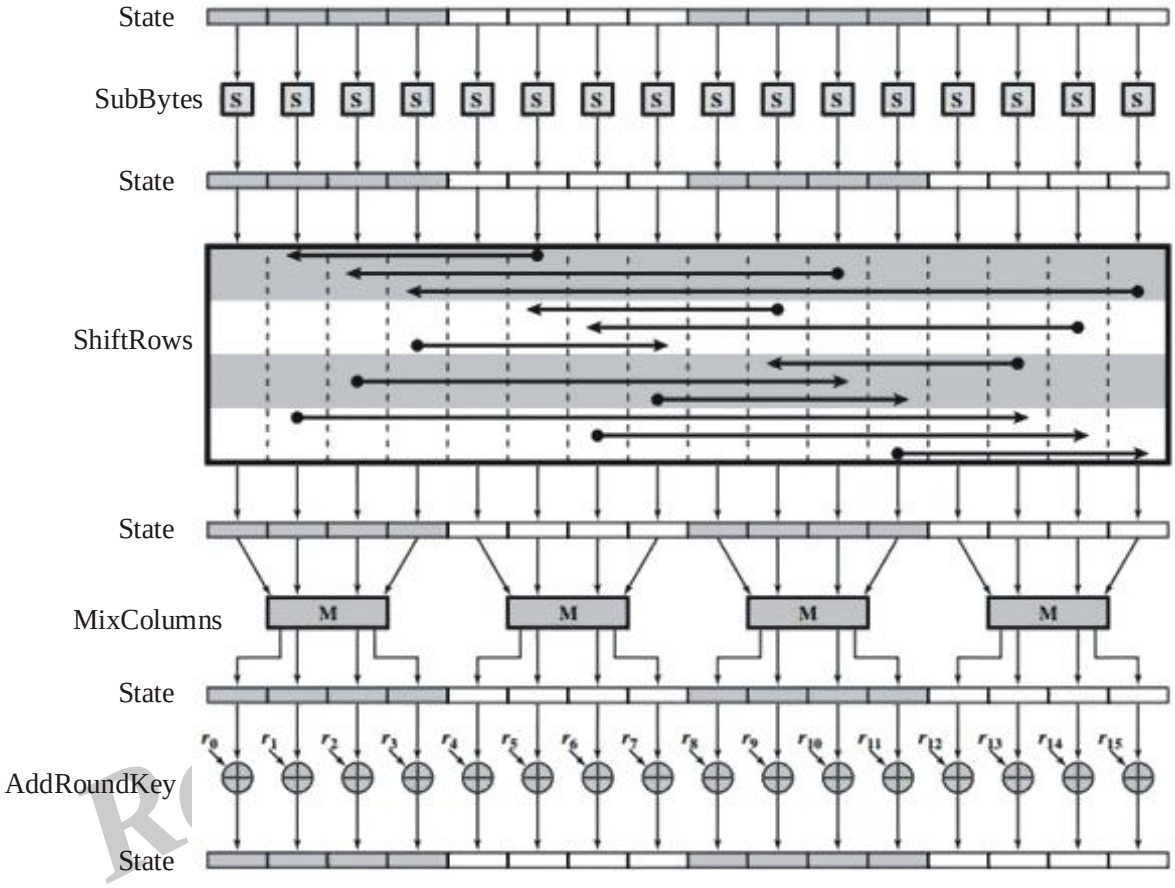


Fig. : AES Encryption Round

Q8. Explain about AES Key Expansion Algorithm.

Ans : **(Imp.)**

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 is filled 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 word whose position in the w array is a multiple of 4, a more complex function is used. Figure illustrates the generation 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
    }
}
```

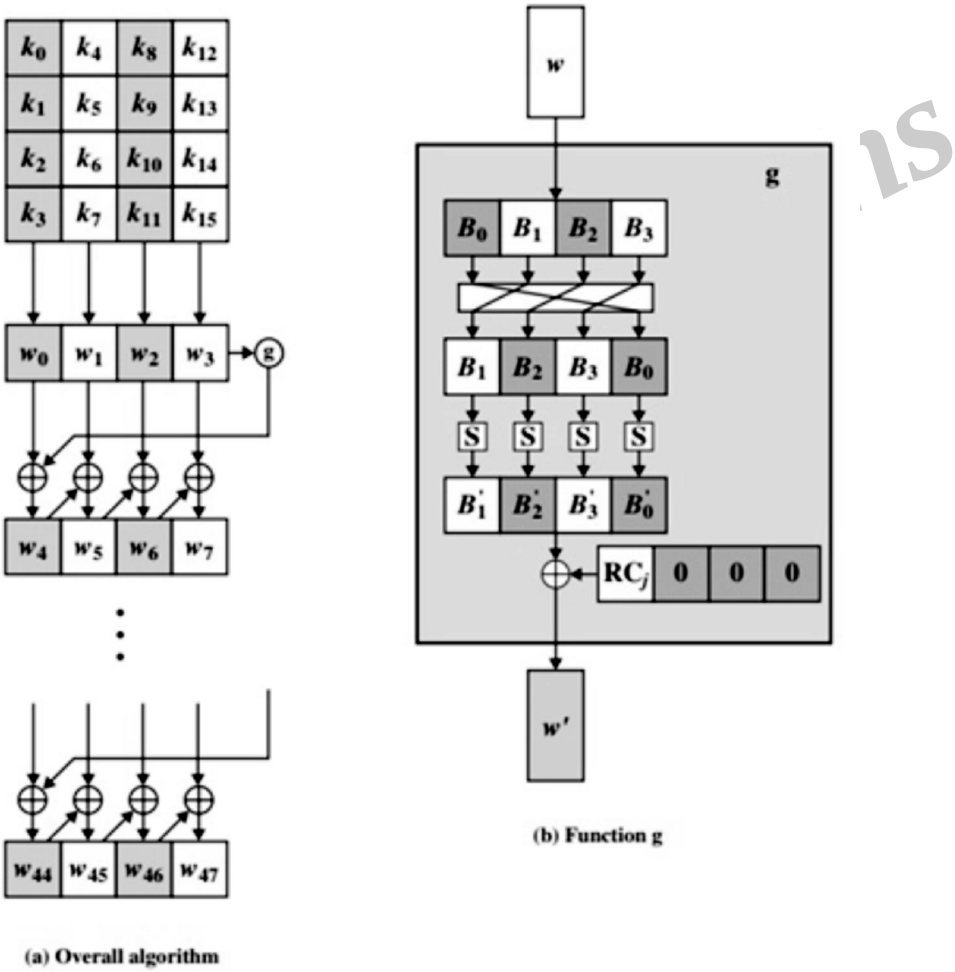


Fig. : AES Key Expansion

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).
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 for each 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

Q9. Illustrate AES with an example.

Ans : **(Imp.)**

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

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

Start of Round	After SubBytes	After ShiftRows	After MixColumns	Round Key
01 89 fe 76				0f 47 0c af
23 ab dc 54				15 d9 b7 7f
45 cd ba 32				71 e8 ad 67
67 ef 98 10				c9 59 d6 98
0e ce f2 d9	ab 8b 89 35	ab 8b 89 35	b9 94 57 75	dc 9b 97 38
36 72 6b 2b	05 40 7f f1	40 7f f1 05	e4 8e 16 51	90 49 fe 81
34 25 17 55	18 3f f0 fc	f0 fc 18 3f	47 20 9a 3f	37 df 72 15
ae b6 4e 88	e4 4e 2f c4	c4 e4 4e 2f	c5 d6 f5 3b	b0 e9 3f a7
65 0f c0 4d	4d 76 ba e3	4d 76 ba e3	8e 22 db 12	d2 49 de e6
74 c7 e8 d0	92 c6 9b 70	c6 9b 70 92	b2 f2 dc 92	c9 80 7e ff
70 ff e8 2a	51 16 9b e5	9b e5 51 16	df 80 f7 c1	6b b4 c6 d3
75 3f ca 9c	9d 75 74 de	de 9d 75 74	2d c5 1e 52	b7 5e 61 c6
5c 6b 05 f4	4a 7f 6b bf	4a 7f 6b bf	b1 c1 0b cc	c0 89 57 b1
7b 72 a2 6d	21 40 3a 3c	40 3a 3c 21	ba f3 8b 07	af 2f 51 ae
b4 34 31 12	8d 18 c7 c9	c7 c9 8d 18	f9 1f 6a c3	df 6b ad 7e
9a 9b 7f 94	b8 14 d2 22	22 b8 14 d2	1d 19 24 5c	39 67 06 c0
71 48 5c 7d	a3 52 4a ff	a3 52 4a ff	d4 11 fe 0f	2c a5 f2 43
15 dc da a9	59 86 57 d3	86 57 d3 59	3b 44 06 73	5c 73 22 8c
26 74 c7 bd	f7 92 c6 7a	c6 7a f7 92	cb ab 62 37	65 0e a3 dd
24 7e 22 9c	36 f3 93 de	de 36 f3 93	19 b7 07 ec	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
	0123456789abcdeffedcba9876543210 0023456789abcdeffedcba9876543210	1
0	0e3634aece7225b6f26b174ed92b5588 0f3634aece7225b6f26b174ed92b5588	1
1	657470750fc7ff3fc0e8e8ca4dd02a9c c4a9ad090fc7ff3fc0e8e8ca4dd02a9c	20
2	5c7bb49a6b72349b05a2317ff46d1294 fe2ae569f7ee8bb8c1f5a2bb37ef53d5	58
3	7115262448dc747e5cdac7227da9bd9c ec093dfb7c45343d689017507d485e62	59
4	f867aee8b437a5210c24c1974cffeabc 43efdb697244df808e8d9364ee0ae6f5	61
5	721eb200ba06206dcbd4bce704fa654e 7b28a5d5ed643287e006c099bb375302	68
6	0ad9d85689f9f77bc1c5f71185e5fb14 3bc2d8b6798d8ac4fe36a1d891ac181a	64
7	db18a8ffa16d30d5f88b08d777ba4eaa 9fb8b5452023c70280e5c4bb9e555a4b	67
8	f91b4fbfe934c9bf8f2f85812b084989 20264e1126b219aef7feb3f9b2d6de40	65
9	cca104a13e678500ff59025f3bafaa34 b56a0341b2290ba7dfdfbddcd8578205	61
10	ff0b844a0853bf7c6934ab4364148fb9 612b89398d0600cde116227ce72433f0	58

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	ff0b844a0853bf7c6934ab4364148fb9 fc8923ee501a7d207ab670686839996b	53

Table : Avalanche Effect in AES: Change in Key

Q10. Explain about the applications of AES.

Ans : **(Imp.)**

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.

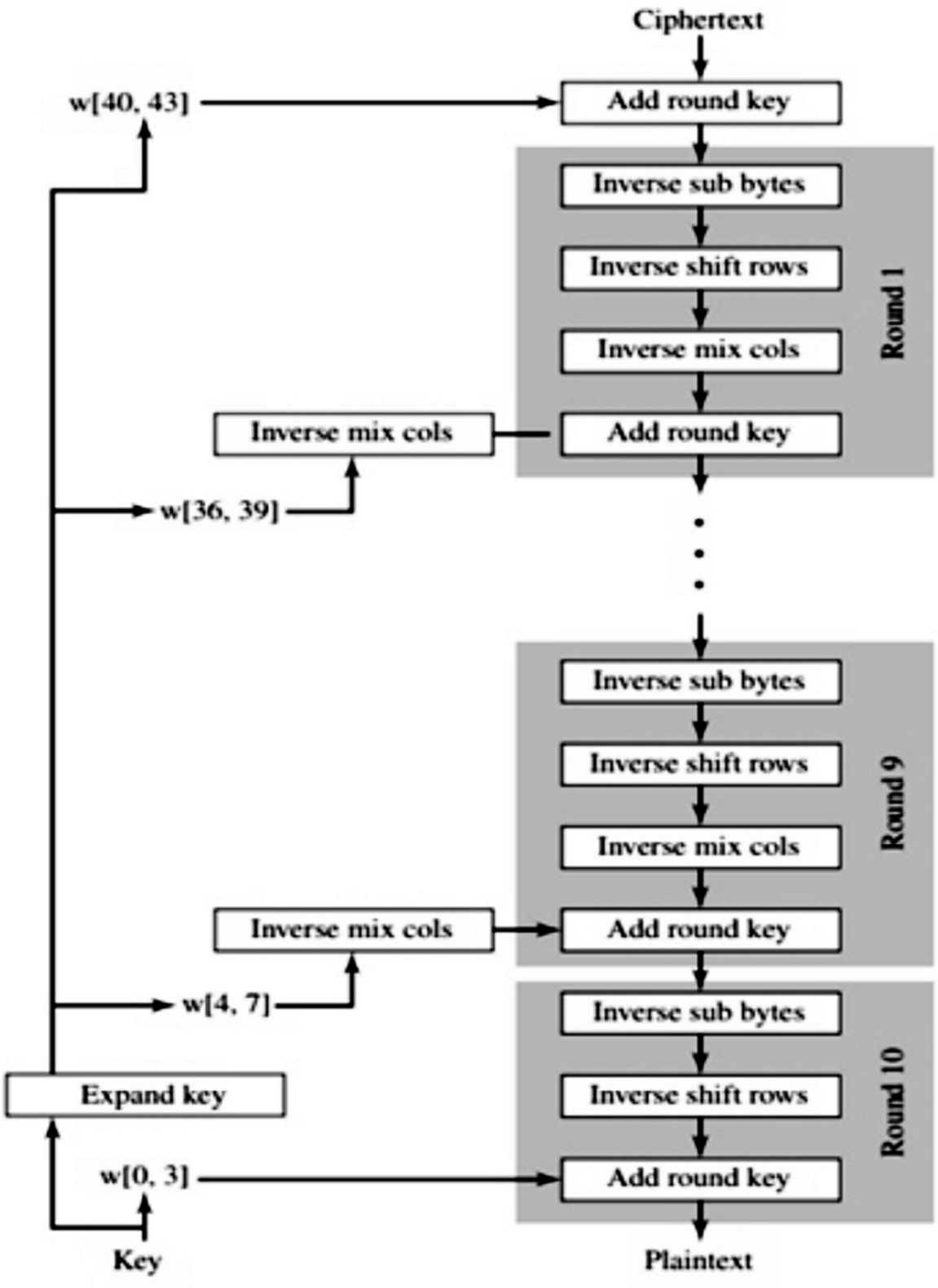


Fig. : illustrates the equivalent decryption algorithm

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 :

$$\begin{aligned}
Tmp &= S_0,j \oplus S_1,j \oplus S_2,j \oplus S_3,j \\
S'_0,j &= S_0,j \oplus Tmp \oplus \left[2 \bullet (S_0,j \oplus S_1,j)\right] \\
S'_1,j &= S_1,j \oplus Tmp \oplus \left[2 \bullet (S_1,j \oplus S_2,j)\right] \qquad (5.9) \\
S'_2,j &= S_2,j \oplus Tmp \oplus \left[2 \bullet (S_2,j \oplus S_3,j)\right] \\
S'_3,j &= S_3,j \oplus Tmp \oplus \left[2 \bullet (S_3j \oplus S_0,j)\right]
\end{aligned}$$

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

$$\begin{aligned}
Tmp &= S_0,j \oplus S_1,j \oplus S_2,j \oplus S_3,j \\
S'_0 &= S_0,j \oplus Tmp \oplus X2\left[S_0,j \oplus S_1,j\right] \\
S'_1,c &= S_1,j \oplus Tmp \oplus X2\left[S_1,j \oplus S_2,j\right] \\
S'_2,c &= S_2,j \oplus Tmp \oplus X2\left[S_2,j \oplus S_3,j\right] \\
S'_3,j &= S_3,j \oplus Tmp \oplus X2\left[S_3,j \oplus S_0,j\right]
\end{aligned}$$

32-Bit Processrr. For a 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.

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

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.

2.1.3 Key Distribution

Q11. Explain about Symmetric Key distribution using Asymmetric Encryption.

Ans : **(Imp.)**

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.

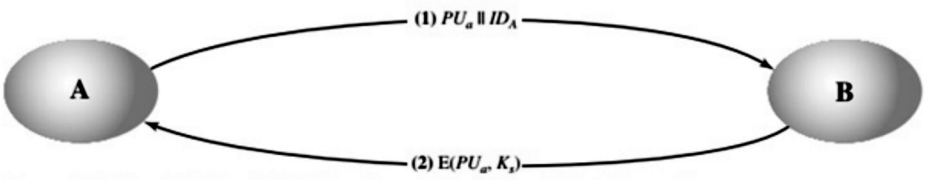


Fig.: Simple Use of Public -Key Encryption to Establish a session Key

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.

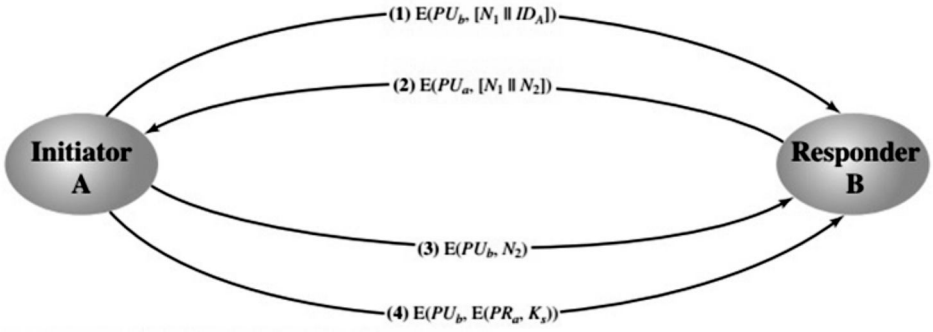


Fig.: Public Key Distribution of Secret Keys

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.

2.2 Public Key Cryptography

Q12. Explain various principles of public key cryptography.

Ans : **(Imp.)**

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.

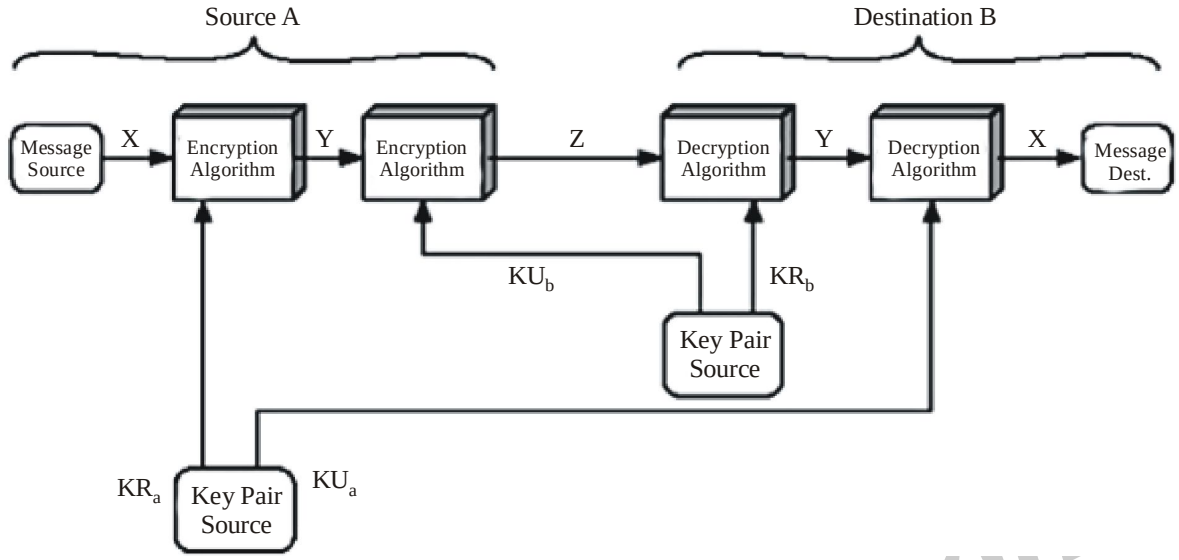


Fig. : Public Key Cryptosystem

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.

2.2.1 RSA

Q13. Explain briefly about RSA Algorithm.

Ans : **(Imp.)**

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:

$C = M^e \bmod n$

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

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

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 $M^{ed} = M \bmod n$ 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:

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

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

$$m^{k\Phi(n)+1} = m^{k(p-1)(q-1)+1} = m \bmod n$$

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:

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

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

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

1. Key Generation

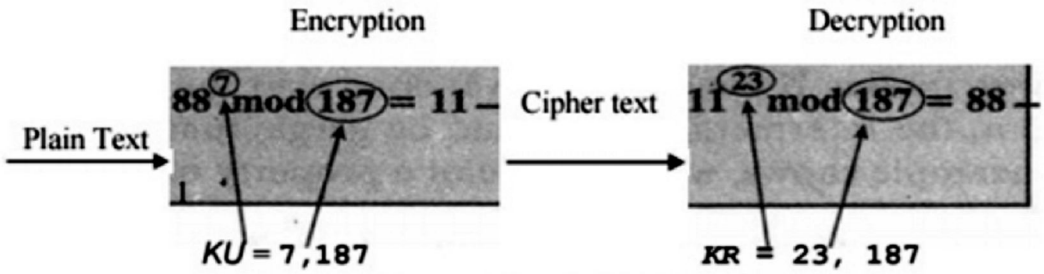
Select p, q		p ,q both prime p≠q
Calculate n = p x q		
Calculate $\phi(n) = (p-1)(q-1)$		
Select integer	e	$\gcd(\phi(n), e) = 1;\ 1 < e < \phi(n)$
Calculate	d	$d = e^{-1} \bmod \phi(n)$
Public key		KU = { e,n}
Private key		KR = {d,n}

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.

2.2.2 ECC

2.2.2.1 Key Exchange

Q14. Explain about Elliptic Curve Cryptography.

Ans : **(Imp.)**

Elliptic curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. The use of elliptic curves in cryptography was suggested independently by Neal Koblitz and Victor S. Miller in 1985. The principal attraction of ECC compared to RSA is that it appears to offer equal security for a far smaller bit size, thereby reducing the processing overhead.

Elliptic Curve over GF(p)

Let GF(p) be a finite field, p > 3, and let a, b

$4a^3 + 27b^2 \equiv 0 \pmod p$. An elliptic curve,

- GF(p)are constant such that

E(a,b)(GF(p)), is defined as the set of points (x,y) C GF(p) * GF(p) which satisfy the equation

$y^2 \equiv x^3 + ax + b \pmod p$, together with a special point, 0, called the point at infinity.

Let P and Q be two points on E(a,b)(GF(p)) and O is the point at infinity.

- P + 0 = 0 + P = P
- If P = (x1,y1) then – P = (x1 ,–y1) and P + (–P) = 0.
- If P = (x1,y1) and Q = (x2,y2), and P and Q are not 0.

then P + Q = (x3 ,y3) where

x3 = λ 2 – x1– x2

y3 = λ (x1 – x3)– y1 and λ = (y2 – y1)/(x2 – x1) if P ≠ Q

λ = (3 × 12 + a)/ 2y1 if P = Q

$$86 \quad \lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \text{ for } \_ x_1 \neq x_2 \\ \frac{3x_1^{\,2} + a}{2y_1} \text{ for } \_ x_1 = x_2 \end{cases}$$

An elliptic curve may be defined over any finite field GF(q). For GF(2m), the curve has a different form:- y2 + xy = x3 + ax2 + b, where b ! = 0.

Cryptography with Elliptic Curves

The addition operation in ECC is the counterpart of modular multiplication in RSA, and multiple addition is the counterpart of modular exponentiation. To form a cryptographic system using elliptic curves, some kind of hard problem such as discrete logarithm or factorization of prime numbers is needed. Considering the equation, Q = kP, where Q,P are points in an elliptic curve, it is "easy" to compute Q given k, P, but "hard" to find k given Q,P. This is known as the elliptic curve logarithm problem. k could be so large as to make brute- force fail.

ECC Key Exchange

Pick a prime number p= 2180 and elliptic curve parameters a and b for the equation $y2 \equiv x3 + ax + b \pmod p$ which defines the elliptic group of points Ep(a,b). Select generator point G = (x1,y1) in Ep(a,b) such that the smallest value for which nG = 0 be a very large prime number. Ep(a,b) and G are parameters of the cryptosystem known to all participants. The following steps take place:

- A & B select private keys nA < n, nB < n
- compute public keys: PA = nA × G, PB = nB × G
- Compute shared key: K = nA × PB, K = nB × PA {same since K = nA × nB × G}

Encrption / Decryption

ECC Encryption/Decryption As with key exchange system, an encryption/decryption system requires a point G and and elliptic group Ep(a,b) as parameters. First thing to be done is to encode the plaintext message m to be sent as an x-y point Pm. Each user chooses private key nA < n and computes public key PA = nA × G. To encrypt and send a message to Pm to B, A chooses a random positive integer k and produces the ciphertext Cm consisting of the pair of points Cm = {kG, Pm + kPb}. here, A uses B's public key. To decrypt the ciphertext, B multiplies the first point in the pair by B's secret key and subtracts the result from the second point Pm + kPb – nB(kG) = Pm + k(nBG) – nB(kG) = Pm A has masked the message Pm by adding kPb to it. Nobody but A knows the value of k, so even though Pb is a public key, nobody can remove the mask kPb. For an attacker to recover the message, he has to compute k given G and kG, which is assumed hard.

Security of ECC To protect a 128 bit AES key it would take a RSA Key Size of 3072 bits whereas an ECC Key Size of 256 bits.

Computational Effort for Cryptanalysis of Elliptic Curve Cryptography Compared to RSA

Key Size	MIPS - Years
150	3.8×10^{10}
205	7.1×10^{18}
234	1.6×10^{28}

(a) Elliptic Curve Logarithms using the Pollard rho Method

Key Size	MIPS - Years
512	3×10^{4}
768	2×10^{8}
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

(b) Integer Factorization using the General Number Field Sieve

Hence for similar security ECC offers significant computational advantages.

Applications

- Wireless communication devices
- Smart cards
- Web servers that need to handle many encryption sessions
- Any application where security is needed but lacks the power storage and computational power that is necessary for our current cryptosystems.

2.2.3 Diffie - Hellman

Q15. Briefly explain about Diffie - Hellman algorithm.

Ans : **(Imp.)**

The first published public-key algorithm appeared in the seminal paper by Diffie and Hellman that defined public-key cryptography [DIFF76b] and is generally referred to as Diffie-Hellman key exchange. A number of commercial products employ this key exchange technique.

The purpose of the algorithm is to enable two users to securely exchange a key that can then be used for subsequent encryption of messages. The algorithm itself is limited to the exchange of secret values.

The Diffie-Hellman algorithm depends for its effectiveness on the difficulty of computing discrete logarithms. Briefly, we can define the discrete logarithm in the following way. First, we define a primitive root of a prime number p as one whose powers modulo p generate all the integers from 1 to p – 1. That is, if a is a primitive root of the prime number p, then the numbers

$$a \bmod p,\ a^2 \bmod p, \ldots,\ a^{p-1} \bmod p$$

are distinct and consist of the integers from 1 through p — 1 in some permutation.

For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$$b \equiv a^i \pmod p \text{ where } 0 \leq i \leq (p - 1)$$

The exponent i is referred to as the discrete logarithm of b for the base a, mod p. We express this value as $dlog_{a,p}$ (b).

For this scheme, there are two publicly known numbers: a prime number q and an integer α that is a primitive root of q. Suppose the users A and B wish to exchange a key. User A selects a random integer $X_A < q$ and computes $Y_A = \alpha^{X_A} \bmod q$. Similarly, user B independently selects a random integer $X_B < q$ and computes $Y_B = \alpha^{X_B} \bmod q$. Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and user B computes the key as $K = (Y_A)^{X_B} \bmod q$. These two calculations produce identical results:

$$K = (Y_B)^{X_A} \bmod q$$

$$= \left(\alpha^{X_B} \bmod q\right)^{X_A} \bmod q$$

$$= (\alpha^{X_B})^{X_A} \bmod q \quad \text{by the rules of modular arithmetic}$$

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

$= (\alpha^{X_A})^{X_B} \bmod q$

$= (\alpha^{X_A} \bmod q)^{X_B} \bmod q$

$= (Y_A)^{X_B} \bmod q$

Global Public Elements	
q	prime number
α	$\alpha < q$ and α a primitive root of q

User A Key Generation	
Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation	
Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Calculation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Calculation of Secret Key by User B

$K = (Y_A)^{X_B} \bmod q$

Fig.:The Diffie - Hellman Key Exchange Algorithm

The result is that the two sides have exchanged a secret value. Furthermore, because X_A and X_B are private, an adversary only has the following ingredients to work with: q, α, Y_A, and Y_B. Thus, the adversary is forced to take a discrete logarithm to determine the key. For example, to determine the private key of user B, an adversary must compute

$$X_B = d\log_{\alpha.q}(Y_B)$$

The adversary can then calculate the key K in the same manner as user B calculates it.

The security of the Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large primes, the latter task is considered infeasible.

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

A computes $Y_A = 3^{97} \bmod 353 = 40$.

B computes $Y_B = 3^{233} \bmod 353 = 248$.

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

A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$.

B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$.

We assume an attacker would have available the following information:

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

In this simple example, it would be possible by brute force to determine the secret key 160. In particular, an attacker E can determine the common key by discovering a solution to the equation $3^a \bmod 353 = 40$ or the equation $3^b \bmod 353 = 248$. The brute-force approach is to calculate powers of 3 modulo 353, stopping when the result equals either 40 or 248. The desired answer is reached with the exponent value of 97, which provides $3^{97} \bmod 353 = 40$.

With larger numbers, the problem becomes impractical.

Key Exchange Protocols

Figure 10.8 shows a simple protocol that makes use of the Diffie-Hellman calculation. Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection. User A can generate a one-time private key X_A, calculate Y_A, and send that to user B. User B responds by generating a private value X_B, calculating Y_B, and sending Y_B to user A. Both users can now calculate the key. The necessary public values q and α would need to be known ahead of time. Alternatively, user A could pick values for q and a and include those in the first message.

As an example of another use of the Diffie-Hellman algorithm, suppose that a group of users (e.g., all users on a LAN) each generate a long-lasting private value X_i (for user i) and calculate a public value Y_i These public values, together with global public values for q and α , are stored in some central directory. At any time, user j can access user i's public value, calculate a secret key, and use that to send an encrypted message to user A. If the central directory is trusted, then this form of communication

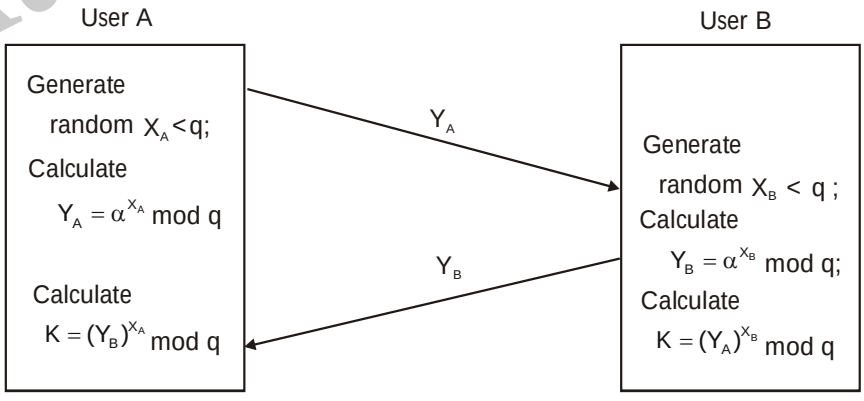


Fig.: Diffie -Hellman Key Exchange

provides both confidentiality and a degree of authentication. Because only i and j can determine the key, no other user can read the message (confidentiality). Recipient i knows that only user j could have created a message using this key (authentication). However, the technique does not protect against replay attacks.

Man-in-the-Middle Attack

The protocol depicted in Figure 10.8 is insecure against a man-in-the-middle attack. Suppose Alice and Bob wish to exchange keys, and Darth is the adversary. The attack proceeds as follows:

1. Darth prepares for the attack by generating two random private keys X_{D1} and X_{D2}, and then computing the corresponding public keys Y_{D1} and Y_{D2}.
2. Alice transmits Y_A to Bob.
3. Darth intercepts Y_A and transmits Y_{DI} to Bob.
   Darth also calculates $K2 = (Y_A)^{X_{D2}} \bmod q$.
4. Bob receives Y_{D1} and calculates $K1 = (Y_{D1})^{X_B} \bmod q$.
5. Bob transmits X_A to Alice.
6. Darth intercepts X_A and transmits Y_{D2} to Alice.
   Darth calculates $K1 = (Y_B)^{X_{D1}} \bmod q$.
7. Alice receives Y_{D2} and calculates $K2 = (Y_{D2})^{X_A} \bmod q$.

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key K1 and Alice and Darth share secret key K2. All future communication between Bob and Alice is compromised in the following way:

1. Alice sends an encrypted message M: E(K2, M).
2. Darth intercepts the encrypted message and decrypts it, to recover M.
3. Darth sends Bob E(K1, M) or E(K1, M'), where M' is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob.

2.2.4 Java Cryptography Extensions

Q16. Explain the concept of JCE ?

Ans : **(Imp.)**

The Java Cryptography Extension (JCE) is an application program interface (API) that provides a uniform framework for the implementation of security features in Java

Java Cryptography Architecture

The Java Cryptography Architecture (JCA) is the name for the internal design of the Java cryptography API.

JCA is structured around some central general purpose classes and interfaces. The real functionality behind these interfaces are provided by providers. Core Classes and Interfaces

The Java cryptography API is divided between the following Java packages:

- java.security
- java.security.cert
- java.security.spec
- java.security.interfaces
- javax.crypto
- javax.crypto.spec
- javax.crypto.interfaces

The core classes and interfaces of these packages are:

- Provider
- SecureRandom
- Cipher
- MessageDigest
- Signature
- Mac
- AlgorithmParameters
- AlgorithmParameterGenerator
- KeyFactory
- SecretKeyFactory
- KeyPairGenerator
- KeyGenerator
- KeyAgreement
- KeyStore
- CertificateFactory
- CertPathBuilder
- CertPathValidator
- CertStore

The most commonly used of these classes are covered throughout the rest of this Java Cryptography tutorial.

Provider

The Provider (java.security.Provider) class is a central class in the Java cryptography API. In order to use the Java crypto API you need a Provider set. The Java SDK comes with its own cryptography provider. If you don't set an explicit cryptography provider, the Java SDK default provider is used. However, this provider may not support the encryption algorithms you want to use. Therefore you might have to set your own cryptography provider.

One of the most popular cryptography providers for the Java cryptography API is called Bouncy Castle. Here is an example that sets a BouncyCastleProvider:

```
import org.bouncycastle.jce.provider.BouncyCastleProvider;
import java.security.Security;
public class ProviderExample {
   public static void main(String[] args) {
      Security.addProvider(new BouncyCastleProvider());
   }
}
```

Cipher

The Cipher (javax.crypto.Cipher) class represents a cryptographic algorithm. A cipher can be used to both encrypt and decrypt data. The Cipher class is explained in more detail in the text on the Java Cipher class, but I will give a brief introduction to the Cipher class in the following sections.

Here is how to create a Java Cipher instance:

```
Cipher cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
```

This example creates a Cipher instance which uses the AES encryption algorithm internally.

The Cipher.getInstance (...) method take a String identifying which encryption algorithm to use, as well as a few other configurations of the algorithm. In the example above, the CBC part is a mode the AES algorithm can work in. The PKCS5Padding part is how the AES algorithm should handle the last bytes of the data to encrypt, if the data does not align with a 64 bit or 128 bit block size boundary. What exactly that means belongs in a tutorial about cryptography in general, not a tutorial about the Java cryptography API.

Initializing the Cipher

Before the Cipher instance can be used it must be initialized. You initialize the Cipher instance by calling its init() method. The init() method takes two parameters:

- Encryption / Decryption cipher mode
- Key

The first parameter specifies whether the Cipher instance should encrypt or decrypt data. The second parameter specifies they key to use to encrypt or decrypt data with.

Here is a Java Cipher.init() example:

```
byte[] keyBytes    = new byte[]{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
String algorithm  = "RawBytes";
SecretKeySpec key = new SecretKeySpec(keyBytes, algorithm);

cipher.init(Cipher.ENCRYPT_MODE, key);
```

Please note that the way the key is created in this example is not secure, and should not be used in practice. This Java cryptography tutorial will describe how to create keys more securely in sections later.

To initialize a Cipher instance to decrypt data you have to use the Cipher.DECRYPT_MODE, like this:

```
cipher.init(Cipher.DECRYPT_MODE, key);
```

Encrypting or Decrypting Data

Once the Cipher is properly initialized you can start encrypting or decrypting data. You do so by calling the Cipher update() or doFinal() methods.

The update() method is used if you are encrypting or decrypting part of a bigger chunk of data. The doFinal() method is called when you are encrypting the last part of the big chunk of data, or if the block you pass to doFinal() represents the complete data block to encrypt.

Here is an example of encrypting some data with the doFinal() method

```
byte[] plainText  = "abcdefghijklmnopqrstuvwxyz".getBytes("UTF-8");
byte[] cipherText = cipher.doFinal(plainText);
```

To decrypt data you would have passed cipher text (encrypted data) into the doFinal() or doUpdate() method instead.

Keys

To encrypt or decrypt data you need a key. There are two types of keys - depending on which type of encryption algorithm you use:

- Symmetric keys
- Asymmetric keys

Symmetric keys are used for symmetric encryption algorithms. Symmetric encryption algorithms use the same key for encryption and decryption.

Asymmetric keys are used for asymmetric encryption algorithms. Asymmetric encryption algorithms use one key for encryption, and another for decryption. The public key - private key encryption algorithms are examples of asymmetric encryption algorithms.

Somehow the party that needs to decrypt data needs to know the key needed to decrypt the data. If the party decrypting the data is not the same as the party encrypting it, somehow these two parties need to agree on a key, or exchange the key. This is referred to as *key exchange*.

Key Security

Keys should be hard to guess so an attacker cannot easily guess the encryption key. The example in the previous section about the Cipher class used a very simple, hardcoded key. This is not a good idea in practice. If they key is easy to guess, it is easy for an attacker to decrypt the encrypted data and possibly create fake messages herself.

It is important to make a key hard to guess. Thus, a key should consist of random bytes. The more random, the better, and the more bytes, the harder to guess because there are more possible combinations.

Generating a Key

You can use the Java KeyGenerator class to generate more random encryption keys. The KeyGenerator is covered in a bit more detail in the text about the Java KeyGenerator, but I will show you an example of how to use it here.

Here is a Java KeyGenerator example:

```
KeyGenerator keyGenerator = KeyGenerator.getInstance("AES");

SecureRandom secureRandom = new SecureRandom();
int keyBitSize = 256;
keyGenerator.init(keyBitSize, secureRandom);

SecretKey secretKey = keyGenerator.generateKey();
```

The resulting SecretKey instance can be passed to the Cipher.init() method, like this:

```
cipher.init(Cipher.ENCRYPT_MODE, secretKey);
```

Generating a Key Pair

Asymmetric encryption algorithms use a key pair consisting of a public key and a private key to encrypt and decrypt data. To generate an asymmetric key pair you can use the KeyPairGenerator (java.security.KeyPairGenerator). The KeyPairGenerator is covered in a bit more detail in Java KeyPairGenerator tutorial, but here is a simple Java KeyPairGenerator example:

```
SecureRandom secureRandom = new SecureRandom();

KeyPairGenerator keyPairGenerator =
KeyPairGenerator.getInstance("DSA");

KeyPair keyPair = keyPairGenerator.generateKeyPair();
```

KeyStore

The Java KeyStore is a database that can contain keys. A Java KeyStore is represented by the KeyStore java.security.KeyStore) class. A KeyStore can hold the following types of keys:

- Private keys
- Public keys + certificates
- Secret keys

Private and public keys are used in asymmetric encryption. A public key can have an associated certificate. A certificate is a document that verifies the identity of the person, organization or device claiming to own the public key. A certificate is typically digitally signed by the verifying party as proof.

Secret keys are used in symmetric encryption.

The KeyStore class is quite advanced so it is described in more detail in its own Java KeyStore Tutorial.

Keytool

The Java Keytool is a command line tool that can work with Java KeyStore files. The Keytool can generate key pairs into a KeyStore file, export certificates from, and import certificates into a KeyStore and several other functions.

The Keytool comes with the Java installation. The Keytool is described in more detail in the tutorial about the Java Keytool.

Mac

The Java Mac class is used to create a MAC from a message. The term MAC is short for Message Authentication Code. A MAC is similar to a message digest, but uses an additional key to encrypt the

message digest. Only by having both the original data and the key can you verify the MAC. Thus, a MAC is a more secure way to guard a block of data from modification than a message digest. The Mac class is described in more detail in the Java Mac tutorial, but below is a short introduction.

You create a Java Mac instance by calling the Mac.getInstance() method, passing as parameter the name of the algorithm to use. Here is how that looks:

```
Mac mac = Mac.getInstance("HmacSHA256");
```

Before you can create a MAC from data you must initialize the Mac instance with a key. Here is an example of initializing the Mac instance with a key:

```
byte[] keyBytes = new byte[]{0,1,2,3,4,5,6,7,8 ,9,10,11,12,13,14,15};
String algorithm = "RawBytes";
SecretKeySpec key = new SecretKeySpec(keyBytes, algorithm);

mac.init(key);
```

Once the Mac instance is initialized you can calculate a MAC from data by calling the update() and doFinal() method. If you have all the data to calculate the MAC for, you can call the doFinal() method immediately. Here is how that looks:

```
byte[] data = "abcdefghijklmnopqrstuvxyz".getBytes("UTF-8");

byte[] macBytes = mac.doFinal(data);
```

If you only have the access to the data in separate blocks, call update() multiple times with the data, and finish off with a call to doFinal(). Here is how that looks:

```
byte[] data  = "abcdefghijklmnopqrstuvxyz".getBytes("UTF-8");
byte[] data2 = "0123456789".getBytes("UTF-8");

mac.update(data);
mac.update(data2);

byte[] macBytes = mac.doFinal();
```

Signature

The Signature (java.security.Signature) class is used to digital sign data. When data is signed a digital signature is created from that data. The signature is thus separate from the data.

A digital signature is created by creating a message digest (hash) from the data, and encrypting that message digest with the private key of the device, person or organization that is to sign the data. The encrypted message digest is called a digital signature.

To create a Signature instance you call the Signature.getInstance(...) method. Here is an example that creates a Signature instance:

```
Signature signature = Signature.getInstance("SHA256WithDSA");
```

UNIT III

Integrity, Authentication and Non-Repudiation : Hash Function (MD5, SHA5), Message Authentication Code (MAC), Digital Signature (RSA, DSA Signatures), Biometric Authentication.

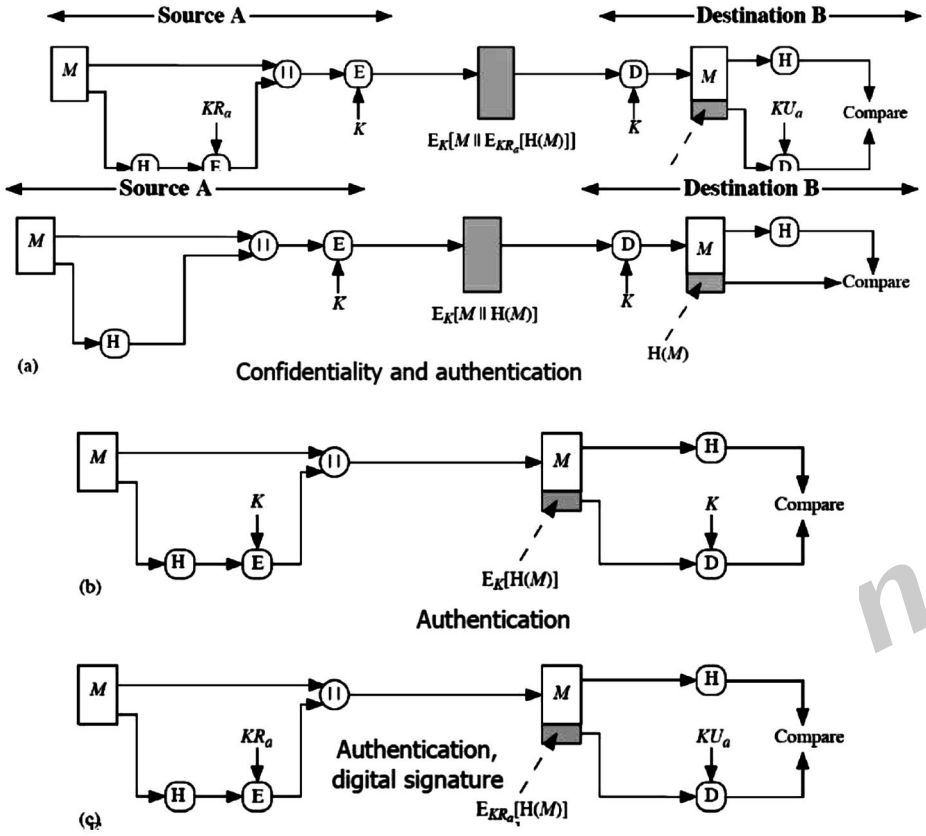
3.1 HASH FUNCTION

Q1. Explain the concept of Hash Function.

Ans : **(Imp.)**

A variation on the message authentication code is the one-way hash function. As with the message authentication code, the hash function accepts a variable-size message M as input and produces a fixed-size hash code H(M), sometimes called a message digest, as output. The hash code is a function of all bits of the message and provides an error-detection capability: A change to any bit or bits in the message results in a change to the hash code. A variety of ways in which a hash code can be used to provide message authentication is shown below and explained stepwise in the table.

$A \rightarrow B : E_X [M \| \| H(M)]$ • Provides confidentiality – Only A and B share K • Provides authentication – H(M) is cryptographically protected (a) Encrypt message plus hash code	$A \rightarrow B : E_{KKa} [H(M)]$ • Provides authentication and digital signature • Provides confidentiality – Only A and B Share K (d) Encrypt result of (c) – shared secret key
$A \rightarrow B : M \| \| E_A [H(M)]$ • Provides authentication – H(M) is cryptographically protected (b) Encrypt hash code - shared secret key	$A \rightarrow B : M \| \| H(M \| \| S)$ • Provides authentication – Only A and B share S (e) Compute hash code of message plus secret
$A \rightarrow B : M \| \| E_{KKa} [H(M)]$ • Provides authentication and digital signature – H(M) is cryptographically protected – Only A could create $E_{KRa}[H(M)]$ (c) Encrypt hash code - secder's private key	$A \rightarrow B : M \| \| H(M \| \| S)$ • Provides authentication – Only A and B share S • Provides confidentiality – Only A and B share K (f) Encrypt result of (c)



Encryption software is quite slow and may be covered by patents. Also encryption hardware costs are not negligible and the algorithms are subject to U.S export control. A fixed-length hash value h is generated by a function H that takes as input a message of arbitrary length: h = H(M).

sends M and H(M)

authenticates the message by computing H(M) and checking the match

Requirements for a hash function

The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. To be used for message authentication, the hash function H must have the following properties can be applied to a message of any size produces fixed-length output

Computationally easy to compute H(M) for any given M

Computationally infeasible to find M such that H(M) = h, for a given h, referred to as the one-way property

Computationally infeasible to find M' such that H(M') = H(M), for a given M, referred to as weak collision resistance.

Computationally infeasible to find M,M' with H(M) = H(M') (to resist to birthday attacks), referred to as strong collision resistance.

Examples of simple hash functions are:

- Bit-by-bit XOR of plaintext blocks: h = D1 ⊕ D2 ⊕ … ⊕ DN
- Rotated XOR –before each addition the hash value is rotated to the left with 1 bit
- Cipher block chaining technique without a secret key.

3.2 MD5

Q2. Explain about MD5.

(OR)

Describe the concept of message degest algorithm.

(OR)

State and explain MD5 algorithm.

Ans : **(Imp.)**

In cryptography, MD5 (Message-Digest algorithm 5) is a widely used, partially insecure cryptographic hash function with a 128-bit hash value. As an Internet standard (RFC 1321), MD5 has been employed in a wide variety of security applications, and is also commonly used to check the integrity of files. An MD5 hash is typically expressed as a 32 digit hexadecimal number. MD5 was designed by Ron Rivest in 1991 to replace an earlier hash function; MD4.In 2007 a group of researchers including Arjen Lenstra described how to create a pair of files that share the same MD5 checksum.

1. MD5 Algorithm Description

We begin by supposing that we have a 1000-bit message as input, and that we wish to find its message digest. The following five steps are performed to compute the message digest of the message.

Step 1. Append Padding Bits

The message is "padded" (extended) so that its length (in bits) is similar to 448, modulo 512. That is, the message is extended so that it is just 64 bits timid of being a multiple of 512 bits long. Padding is always performed, even if the length of the message is already similar to 448, modulo 512.Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448, modulo 512. In all, atleast one bit and at most 512 bits are appended.

Step 2. Append Length:

A 64-bit representation of 1000 (The message length excluding padded one) is appended to the result of the previous step. In the unlikely event that the message length is greater than 2 ^ 64, then only the low-order 64 bits of b are used. At this point the resulting message (that is message + padding + length) has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words.

Step 3: Divide the input into 512-bit blocks:

Now, we divide the input message into blocks, each of length 512 bits.

Step 4. Initialize MD Buffer/Chaining Variables

A four-word buffer (A, B, C, D) is used to compute the message digest. Here each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, loworder bytes first):

A: 01 23 45 67

B: 89 ab cd ef

C: fe dc ba 98

D: 76 54 32 10

Step 5. Process Message in 16-Word Blocks:

Copy the four chaining variables into four corresponding variables a, b, c, and d. The Algorithm considers the combination of abcd as a 128-bit single registers. This is useful for holding intermediate as well as final results.

Divide the current 512-bit block into 16 sub blocks of 32-bit each.

Now we have 4 rounds. In each round, we process all the 16 sub-blocks.

The inputs to each round are:

All the 16 sub-blocks. Say M[0] to M[15] of 32 bits.

The variables a, b, c and d of 32 bits.

Some constants t, an array of 64 elements. Say t[1] to t[64].Since there are four rounds, we use 16 out of the 64 values of t in each round.

The process of rounds:

A process P is first performed on b, c and d. This process P is different in all the four rounds.

The variable a is added to the output of the process P.

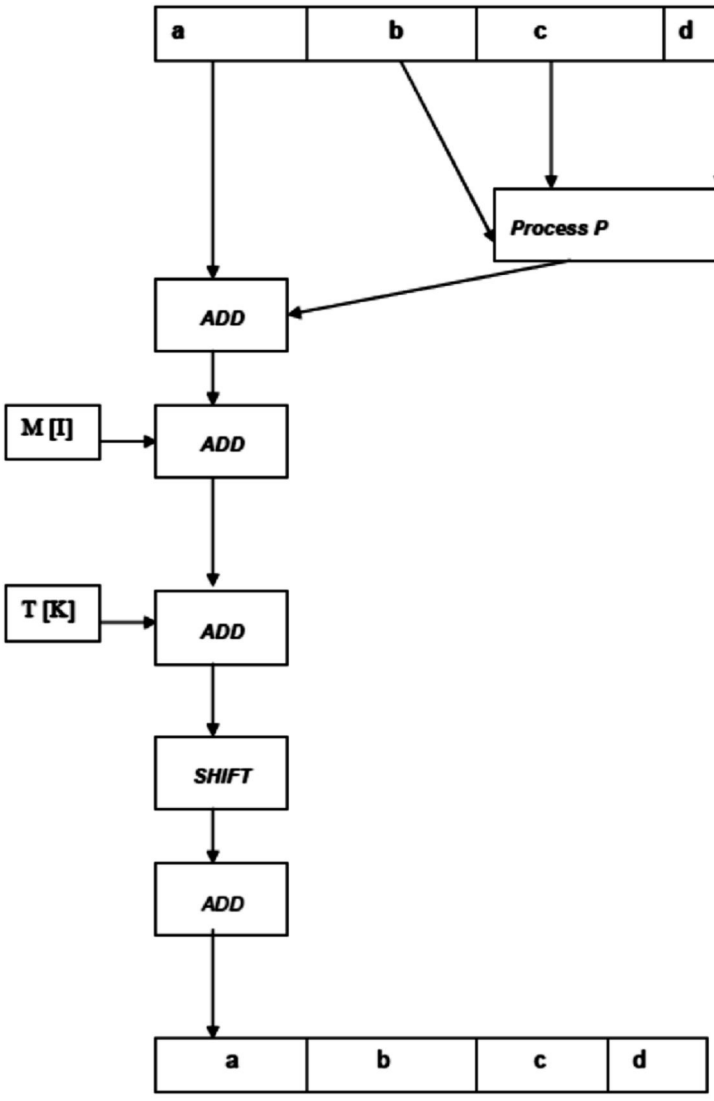
The message sub-block M[I] is added to the output of step 2.

The constant t[k] is added to the output of step 3.

The output of step 4 is circular-left shifted by s bits. The value of s keeps changing. The variable b is added to the output of step 5.

The output of step 6 becomes the new abcd for the next round.

One MD5 Operation:



We define four auxiliary functions that is Process P in our context, that each take as input of three 32-bit words and produce as output one 32-bit word.

Round 1 = (b and c) or (not (b)) and d
Round 2 = (b and d) or (c and (not(c)))
Round 3 = b xor c xor d
Round 4 = c xor (b or not (d))

The MD5 message-digest algorithm is simple to implement, and provides a "fingerprint" or message digest of a message of arbitrary length. It is conjectured that the difficulty of coming up with two messages having the same message digest is on the order of 2 ^ 64 operations, and that the difficulty of coming up with any message having a given message digest is on the order of 2 ^ 128 operations. The MD5 algorithm has been carefully scrutinized for weaknesses. It is, however, a relatively new algorithm and further security analysis is of course justified, as is the case with any new proposal of this sort.

3.3 SHA5

Q3. What is secure hash algorithm? Explain.

Ans : **(Imp.)**

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.

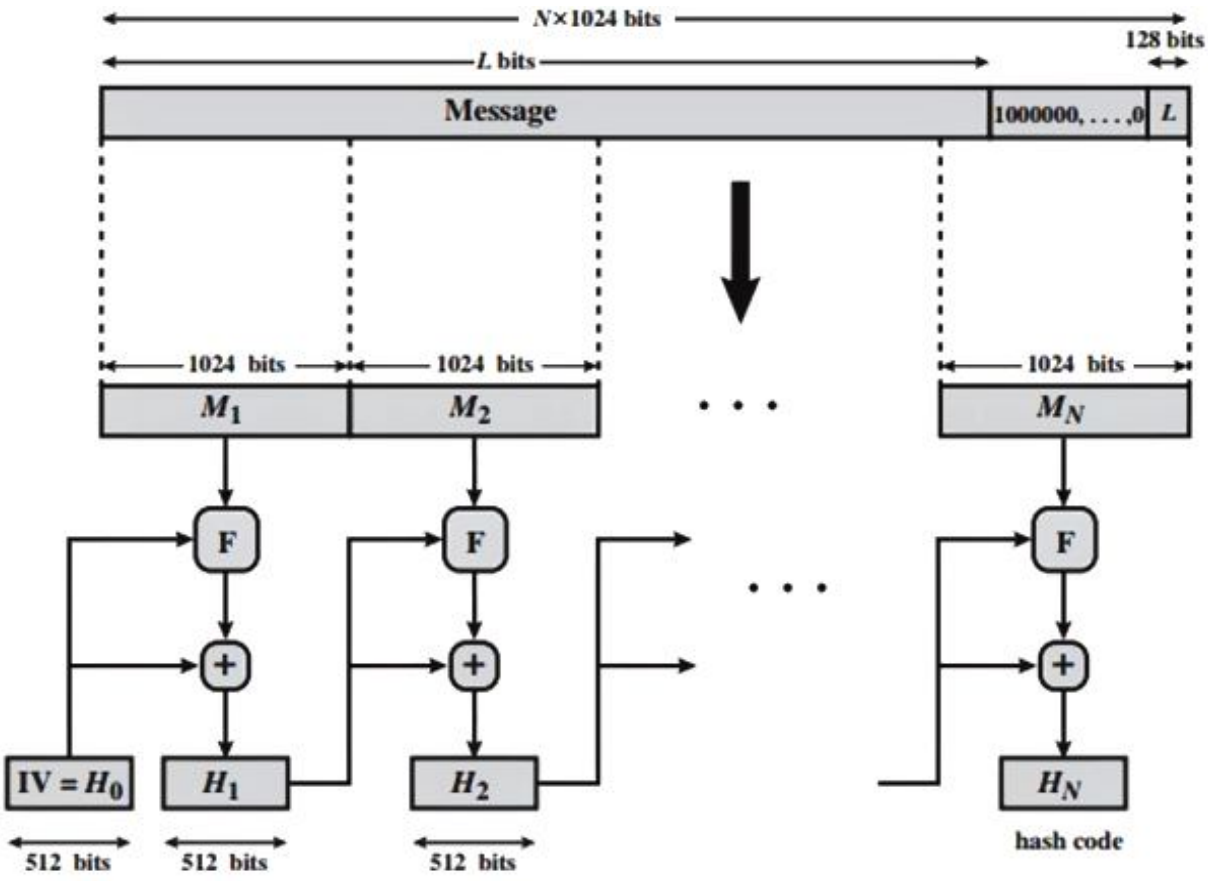
SHA-512 Logic

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 depicts the overall processing of a message to produce a digest. The processing consists of the following steps.

	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	515	515	1024	1024
Word Size	32	32	32	64	64
Number of Step	80	64	64	80	80

Note: All sizes are measured in bits

Table : Comparison of SHA Parameters



+ = word by word addition mod 2^{44}

Fig 1. : 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 Figure1, the expanded message is represented as the sequence of 1024-bit blocks M1, M2, ….. , 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 1. The logic is illustrated in Figure 2.

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 Nth stage is the 512-bit message digest.

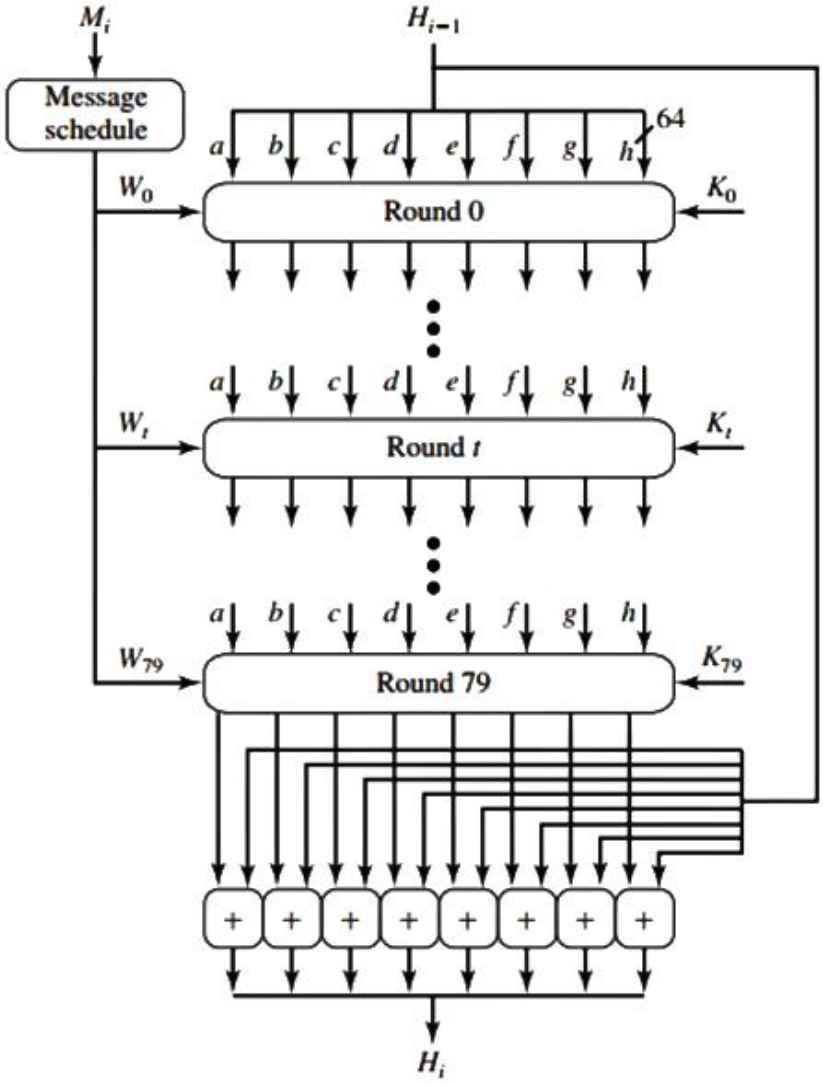


Figure 2. : SHA-512 Processing of a Single 1024-Bit Block

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 MESSAGE AUTHENTICATION CODES(MAC)

Q4. 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.

Q5. 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.

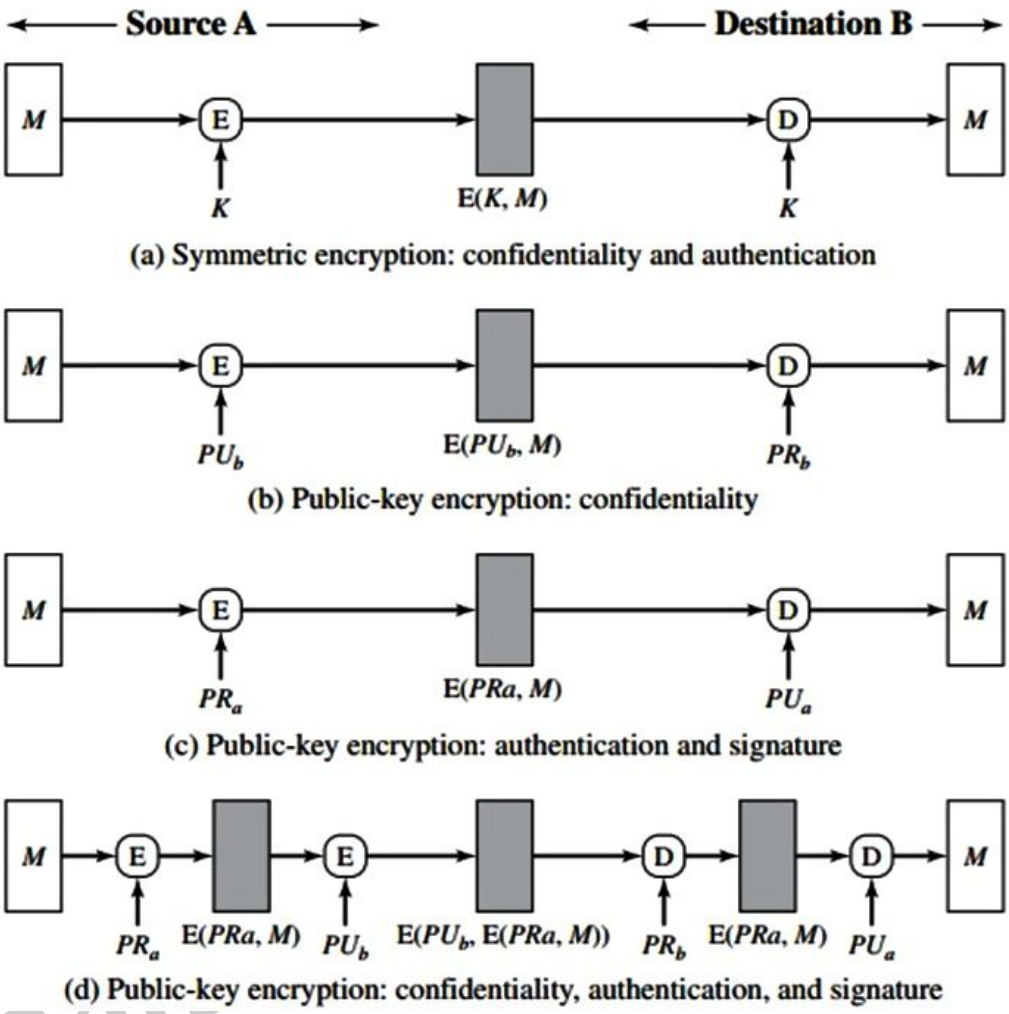


Fig.: Basic Uses of Message Encryption

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.

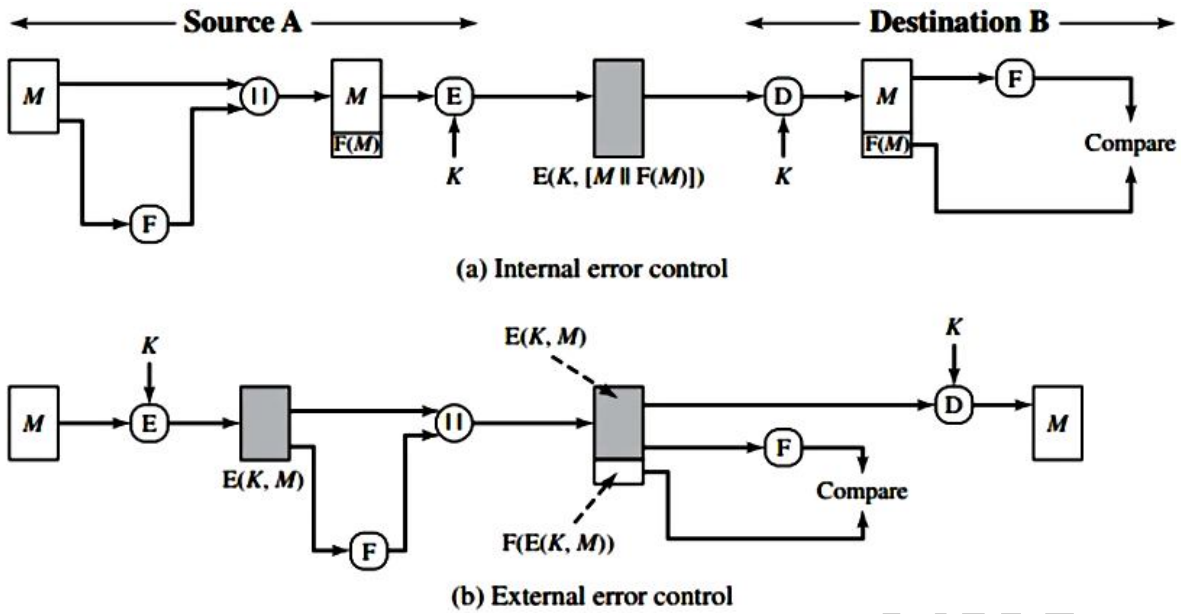


Fig.: Internal and External Error Control

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.

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

Where,

M = input message

C = MAC function

K = shared secret key

MAC = message authentication code

The message plus MAC are transmitted to the intended recipient. The recipient 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.

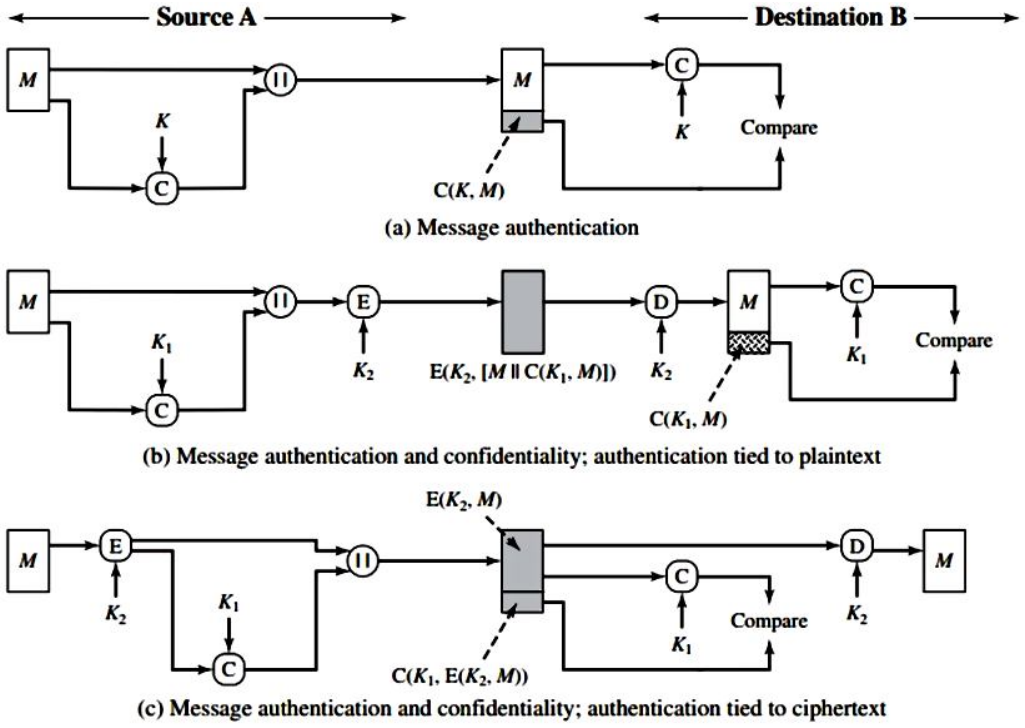


Fig.: Basic Uses of Message Authentication Code (MAC)

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.

Q6. What are the requirements for message authentication codes? Explain briefly.

Ans : **(Imp.)**

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

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

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 $\approx 2^{(k-2\times n)}$

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.

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

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

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

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

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

Q7. Explain about HMAC Alogrithm.

Ans :

HMAC stands for Hash-based Message Authentication Code. HMAC has been chosen as a mandatory security implementation for the Internet Protocol (IP) security, and is also used in the Secure Layer(SSL) protocol, widely used on the Internet.

The fundamental idea behind HMAC is to reuse the existing message digest algorithms, such as MD5 or SHA-1. Obviously, there is no point in reinventing the wheel. Therefore, what HMAC does it to work with any message digest algorithm? That is, it treats the message digest as a black box. Additionally, it uses the shared symmetric key to encrypt the message digest, which produces the output MAC.

HMAC Design Objectives

RFC 2104 lists the following design objectives for HMAC:

- To use, without modifications, available hash functions. In particular, hash functions that perform well in software, and for which code is freely and widely available.
- To allow for easy replaceability of the embedded hash function in case faster or more secure hash functions are found or required.

- To preserve the original performance of the hash function without incurring a significant degradation.
- To use and handle keys in a simple way.
- To have a well understood cryptographic analysis of the strength of the authentication mechanism based on reasonable assumptions about the embedded hash function.

The first two objectives are important to the acceptability of HMAC. HMAC treats the hash function as a "black box." This has two benefits. First, an existing implementation of a hash function can be used as a module in implementing HMAC. In this way, the bulk of the HMAC code is prepackaged and ready to use without modification. Second, if it is ever desired to replace a given hash function in an HMAC implementation, all that is required is to remove the existing hash function module and drop in the new module. This could be done if a faster hash function were desired. More important, if the security of the embedded hash function were compromised, the security of HMAC could be retained simply by replacing the embedded hash function with a more secure one (e.g., replacing SHA with Whirlpool).

The last design objective in the preceding list is, in fact, the main advantage of HMAC over other proposed hash-based schemes. HMAC can be proven secure provided that the embedded hash function has some reasonable cryptographic strengths. We return to this point later in this section, but first we examine the structure of HMAC.

Step 1: Make the length of K equal to b

The algorithm starts with three possibilities, depending on the length of the key K:

- **Length of K < b**

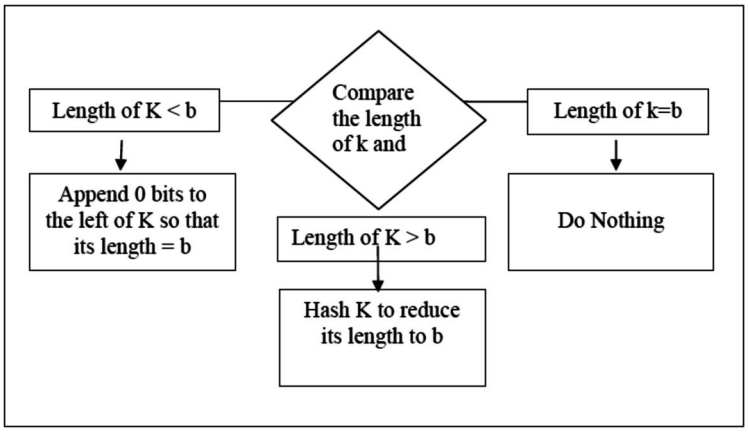
  In this case, we need to expand the key K to make the length of K equal to the number of bits In the original message block (i.e. b). For this, we add as many as 0 bits as required to the left of K, For example, if the initial length of K = 170bits, and b = 512, then add 342 bits, all with a value 0, to the left of K, We shall continue to cal this modified key as K.

- **Length of K = b**

  In this case, we do not take any action, and proceed to step 2.

- **Length of K > b**

  In this case, we need to trim K to make the length of K equal to the number of bits in the original message block (i.e. b). For this, we pass K through the message digest algorithm (H) selected for this particular instance of HMAC, which will give us a key K, trimmed so that its length is equal to b.



Step 1 of HMAC

Step 2: XOR K with ipad to produce S1

We XOR K (i.e. output of Step 1) and ipad to produce a variable called as S1.

Step 3: Append M to S1

We now take the original message (M) and simply append it to the end of S1 (which was calculated in Step 2).

Step 4: Message digest algorithm

Now the selected message digest algorithm (e.g. MD5, SHA-1, etc.) is applied to the output of Step 3 (i.e. to the combination of S1 and M). Let us call the output of this operation as H. This is shown in Fig.

Step 5 : XOR K with opad to produce S2

Now, we XOR K (the output of Step1) with opad to produce a variable called S2, This is shown in Fig.

Step 6: Append H to S2

In this step, we take the message digest calculated in step 4 (i.e. H) and simply append it to the end of S2 (which was calculated in Step 5).

Step 7: Message digest algorithm

Now, the selected message digest algorithm (e.g. MD5, SHA-1, etc.) is applied to the output of Step 6(i.e. to the concatenation of S2 and H). This is the final MAC that we want.

Disadvantages

It appears that the MAC produces by the HMAC algorithm fulfills our requirements of a digital signature. Form a logical perspective, firstly we calculate a fingerprint (message digest) of the original message, and then encrypt it with a symmetric key, which is known only to the sender and the receiver. This gives sufficient confidence to the receiver that the message came from the correct sender only and also that it was not altered during the transit. However, if we observe the HMAC scheme carefully, we will realize that it does not solve our entire problem. What are these problems?

1. We assume in HMAC that the sender and the receiver only know about the symmetric key. However, we have studied in great detail that the problem of symmetric key exchange is quite serious, and cannot be solved easily. The same problem of key exchange is presently the case of HMAC.
2. Even if we assume that somehow the key exchange problem is resolved. HMAC cannot be used if the number of receivers is greater than one. This is because, to produce a MAC by using HMAC, we need to make use of a symmetric key. The symmetric key is supposed to be shared only by two parties: one sender and one receiver. Of course, this problem can be solved if multiple parties (one sender and all the receivers) share the same symmetric key. However, this resolution leads to a third problem.
3. The new problem is that how does a receiver know that the message was prepared and send by the sender, and not by the other receivers? After all, all the co-receivers also know the symmetric key. Therefore, it is quite possible that one of the co-receivers might have created a false message on behalf of the alleged sender, and using HMAC; the co-receiver might have prepared a MAC for the message, and sent the message and the MAC as if it originated at the alleged sender. There is no way to prevent or detect this.
4. Even if we somehow solve the above problem, one major concern remains. Let us go back to our simple case of one sender and one receiver. Now, only two parties the sender (say A, a bank customer) and the receiver (say B, a bank) share the symmetric key secret. Suppose that one fine

day, B transfers al the balance standing in the account of A to a third person's account, and closes A's bank account. A is shocked, and files a suit against B. In the court of law, B argues that A had sent an electronic message in order t perform this transaction, and produces that message as evidence. A claims that she never sent that message, and that it is a forged message. Fortunately, message produced by B as evidence also contained MAC, which was produced on the original message. As we know, only encrypting the message digests of the original message with the symmetric key shared by A and B could have produced it and here is where the trouble is. Even though we have a MAC, how in the world are we now going to prove that the MAC was produced by A or by B? After all, both know about the shared secret key. It is equally possible for either of them to have created the original message and the MAC.

As it turns out, even if we are able to somehow resolve the first three problems, we have no solution for the fourth problem.

Therefore, we cannot trust HMAC to be used in digital signatures.

Q8. Explain about Cipher Message Authentication code(CMAC).

Ans : **(Imp.)**

The Data Authentication Algorithm defined in FIPS PUB 113, also known as the CBC-MAC (cipher block chaining message authentication code). This cipher-based MAC has been widely adopted in government and industry. [BELL00] demonstrated that this MAC is secure under a reasonable set of security criteria, with the following restriction. 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 $X||(X \oplus T)$ since this is once again T.

Black and Rogaway [BLACOO] demonstrated that this limitation could be overcome using three keys: one key of length k to be used at each step of the cipher block chaining and two keys of length n, where k is the key length and n is the cipher block length. This proposed construction was refined by Iwata and Kurosawa so that the two n-bit keys could be derived from the encryption key, rather than being pro-vided separately [IWAT03], This refinement has been adopted by NIST cipher-based message authentication code (CMAC) mode of operation, for use with AES and triple DES. It is specified in NIST Special Publication 800-38B.

First, let us consider 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 K_v For AES, the key size k is 128,192, or 256 bits; for triple DES, the key size is 112 or 168 bits. CMAC is calculated as follows (Figure.):

$C_1 = E(K, MO$

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

$C_3 = E(K, [M_3 \oplus C_2])$

•

•

•

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

$$T = MSB_{Then}(C_n)$$

where

T = message authentication code, also referred to as the tag

Tien = bit length of T

$MSB_s(X)$ = the s leftmost bits of the bit string X

If the message is not an integer multiple of the cipher block length, then the final block is padded to the right (least significant bits) with a 1 and as many Os 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_t.

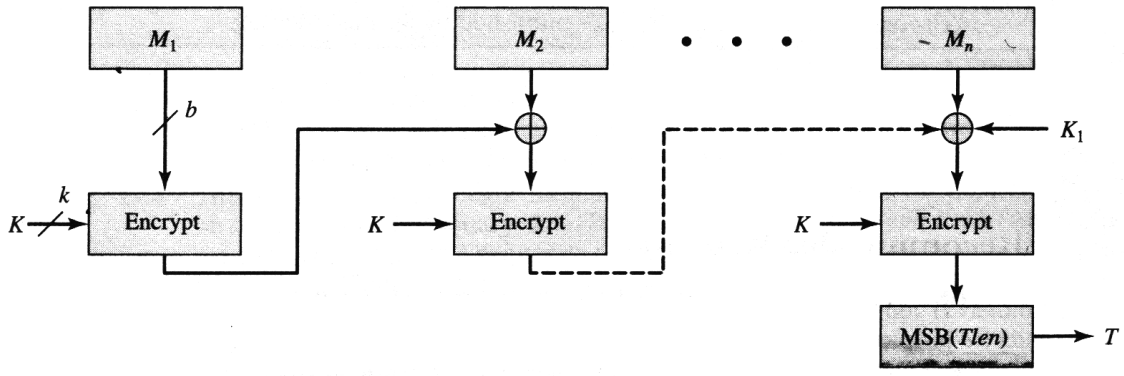
The two /7-bit keys are derived from the k-bit encryption key as follows:

$$L = E(K, 0")$$

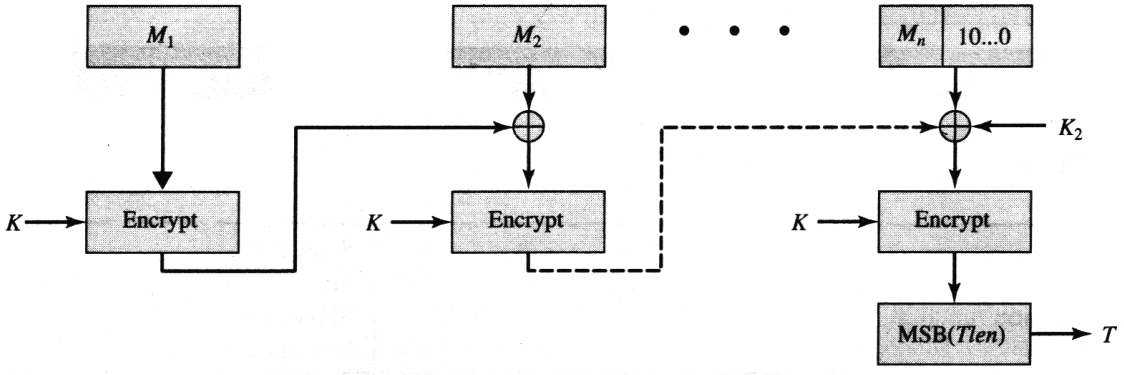
$$K_1 = L^* x$$

$$K_2 = L^* x^2 = (L * x)^* x$$

where multiplication (*) is done in the finite field GF (2^n) and x and x^2 are first and second order polynomials that are elements of GF(2^n). Thus the binary representation of x consists of n – 2 zeros followed by 10; the binary representation of x^2



(a) Message length is integer multiple of block size



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

Fig.: Cipher-Based Message Authentication Code(CMAC)

consists of n – 3 zeros followed by 100. The finite field is defined with respect to an irre-ducible 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 $x^{64} + x^4 + x^3 + x + 1$ and $x^{128} + x^7 + x^2 + x + 1$.

To generate K_1 and K_2, the block cipher is applied to the block that consists entirely of 0 bits. The first subkey is derived from the resulting cipher text 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.

3.5 Digital Signature

Q9. Explain the concept of Digital Signature.

Ans : **(Imp.)**

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

The digital signature is analogous to the handwritten signature. It must have the following properties:

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

Thus, the digital signature function includes the authentication function.

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

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

A secure hash function, embedded in a scheme satisfies these requirements.

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

(i) Direct Digital Signature

The direct digital signature involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source. A digital signature may be formed by encrypting the entire message with the sender's private key fig. or by encrypting a hash code of the message with the sender's private key fig.

Confidentiality can be provided by further encrypting the entire message plus signature with either the receiver's public key (public-key encryption) or a shared secret key (symmetric encryption); for example, see fig. and 11.5d. Note that it is important to perform the signature function first and then an outer confidentiality function. In case of dispute, some third party must view the message and its signature. If the signature is calculated on an encrypted message, then the third party also needs access to the decryption key to read the original message. However, if the signature is the inner operation, then the recipient can store the plaintext message and its signature for later use in dispute resolution.

All direct schemes described so far share a common weakness. The validity of the scheme depends on the security of the sender's private key. If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature. Administrative controls relating to the security of private keys can be employed to thwart or at least weaken this ploy, but the threat is still there, at least to some degree. One example is to require every signed message to include a timestamp (date and time) and to require prompt reporting of compromised keys to a central authority.

Another threat is that some private key might actually be stolen from X at time T. The opponent can then send a message signed with X's signature and stamped with a time before or equal to T.

(ii) Arbitrated Digital Signature

The problems associated with direct digital signatures can be addressed by using an arbiter.

As with direct signature schemes, there is a variety of arbitrated signature schemes. In general terms, they all operate as follows. Every signed message from a sender X to a receiver Y goes first to an arbiter A, who subjects the message and its signature to a number of tests to check its origin and content. The message is then dated and sent to Y with an indication that it has been verified to the satisfaction of the arbiter. The presence of A solves the problem faced by direct signature schemes: that X might disown the message.

The arbiter plays a sensitive and crucial role in this sort of scheme, and all parties must have a great deal of trust that the arbitration mechanism is working properly. The use of a trusted system, described in might satisfy this requirement.

Table based on scenarios described in [AKL83] and [MITC92], gives several examples of arbitrated digital signatures. In the first, symmetric encryption is used. It is assumed that the sender X and the arbiter A share a secret key K_{xa} and that A and Y share secret key K_{ay}. X constructs a message M and computes its hash value H(M). Then X transmits the message plus a signature to A. Hie signature consists of an identifier ID_X of X plus the hash value, all encrypted using K_{xa}. A decrypts the signature and checks the hash value to validate the message. Then A transmits a message to Y.

$(1)\ X \rightarrow A : M \,||\, E(K_{xa}, [ID_X \,||\, H(M)])$
$(2)\ A \rightarrow Y : E(K_{ay}, [ID_X \,||\, M \,||\, E(K_{xa}, [ID_X \,||\, H(M)]) \,||\, T]$

(a) Conventional Encryption, Arbiter Sees Message

$(1)\ X \rightarrow A : ID_X \,||\, E(K_{xy}, M) \,||\, [ID_X \,||\, H(E(K_{xy}, M))])$
$(2)\ A \rightarrow Y : E(K_{ay}, [ID_X \,||\, E(K_{xy}, M)] \,||\, E(K_{xa}, [ID_X \,||\, H(E(K_{xy}, M)) \,||\, T])$

(b) Conventional Encryption, Arbiter Does Not See Message

$(1)\ X \rightarrow A : ID_X \,||\, E(PR_{xy}[ID_X \,||\, E(PU_y, E(PR_x, M))])$
$(2)\ A \rightarrow Y : E(PR_a, [ID_x \,||\, E(PU_y, E(PR_s, M)) \,||\, T])$

(c) Public-Key Encryption, Arbiter Does Not See Message

Table.: Arbitrated Digital Signature Techniques

Notation:

X = sender M = message

Y = recipient T = timestamp

A = Arbiter

encrypted with K_{ay}. The message includes ID_X, the original message from X, the signature, and a timestamp. Y can decrypt this to recover the message and the signature. The timestamp informs Y that this message is timely and not a replay. Y can store M and the signature. In case of dispute, Y, who claims to have received M from X, sends the following message to A:

$$E(K_{ay},\ [ID_X||M||E(K_{xa},\ [ID_X||H(M)])])$$

The arbiter uses K_{ay} to recover ID_X, M, and the signature, and then uses K_{xa} to decrypt the signature and verify the hash code. In this scheme, Y cannot directly check X's signature; the signature is there solely to settle disputes. Y considers the message from X authentic because it comes through A. In this scenario, both sides must have a high degree of trust in A:

- X must trust A not to reveal K_{xa} and not to generate false signatures of the form

  $E(K_{xa},\ [ID_X||H(M)])$.

- Y must trust A to send $E(K_{ay}, [ID_X||M||E(K_{xa}, [ID_X||H(M)])||7])$ only if the hash value is correct and the signature was generated by X.

- Both sides must trust A to resolve disputes fairly.

If the arbiter does live up to this trust, then X is assured that no one can forge his signature and Y is assured that X cannot disavow his signature.

The preceding scenario also implies that A is able to read messages from X to Y and, indeed, that any eavesdropper is able to do so. Table shows a scenario that provides the arbitration as before but also

assures confidentiality. In this case it is assumed that X and Y share the secret key K_{xy}. Now, X transmits an identifier, a copy of the message encrypted with K_{xy}, and a signature to A. The signature consists of the identifier plus the hash value of the encrypted message, all encrypted using K_{xa}. As before, A decrypts the signature and checks the hash value to validate the message. In this case, A is working only with the encrypted version of the message and is prevented from reading it. A then transmits everything that it received from X, plus a timestamp, all encrypted with K_{ay}, to Y.

Although unable to read the message, the arbiter is still in a position to prevent fraud on the part of either X or Y. A remaining problem, one shared with the first scenario, is that the arbiter could form an alliance with the sender to deny a signed message, or with the receiver to forge the sender's signature.

All the problems just discussed can be resolved by going to a public-key scheme, one version of which is shown in table. In this case, X double encrypts a message M first with X's private key, PR_x, and then with Y's public key, PU_y. This is a signed, secret version of the message. This signed message, together with X's identifier, is encrypted again with PR_x and, together with ID_x, is sent to A. The inner, double-encrypted message is secure from the arbiter (and everyone else except Y). However, A can decrypt the outer encryption to assure that the message must have come from X (because only X has PR_x). A checks to make sure that X's private/public key pair is still valid and, if so, verifies the message. Then A transmits a message to Y, encrypted with PR_a. The message includes ID_x, the double-encrypted message, and a timestamp.

This scheme has a number of advantages over the preceding two schemes. First, no information is shared among the parties before communication, preventing alliances to defraud. Second, no incorrectly dated message can be sent, even if PR_x is compromised, assuming that PR_a is not compromised. Finally, the content of the message from X to Y is secret from A and anyone else. However, this final scheme involves encryption of the message twice with a public-key algorithm. We discuss more practical approaches subsequently.

3.5.1 RSA

Q10. Explain the concept of RSA.

Ans :

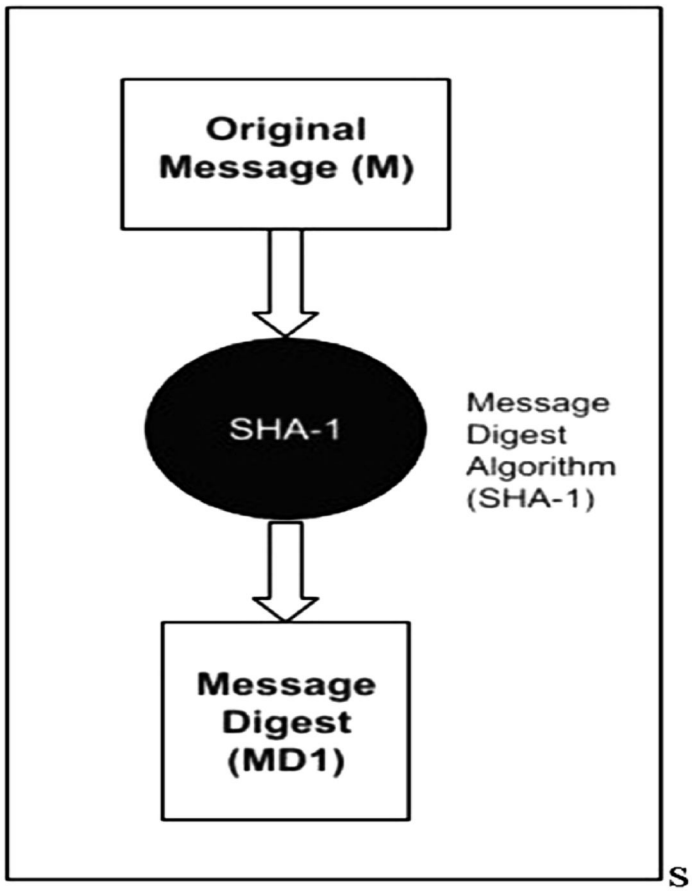
It is the most popular asymmetric cryptographic algorithm. It is primarily used for encrypting message s but can also be used for performing digital signature over a message.

Let us understand how RSA can be used for performing digital signatures step-by-step.

Assume that there is a sender (A) and a receiver (B). A wants to send a message (M) to B along with the digital signature (DS) calculated over the message.

Step-1 :

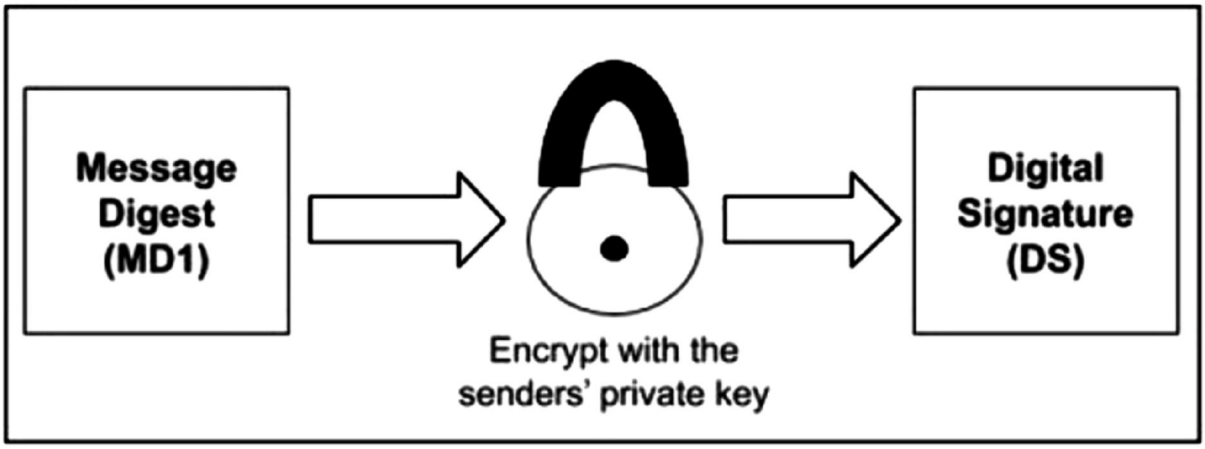
Sender A uses SHA-1 Message Digest Algorithm to calculate the message digest (MD1) over the original message M.



Message digest calculation

Step-2 :

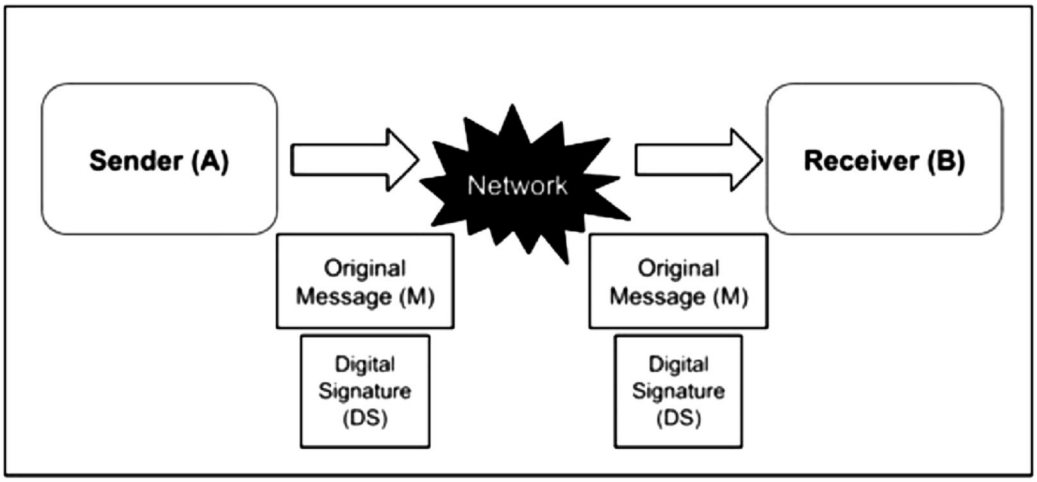
A now encrypts the message digest with its private key. The output of this process is called Digital Signature (DS) of A.



Digital signature creation

Step-3 :

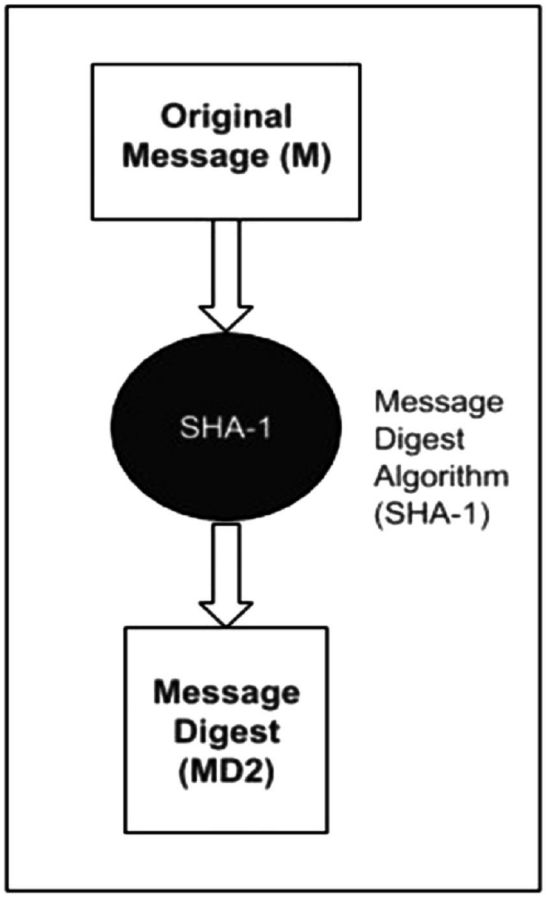
Now sender A sends the digital signature (DS) along with the original message (M) to B.



Transmission of original message and digital signature simultaneously

Step-4 :

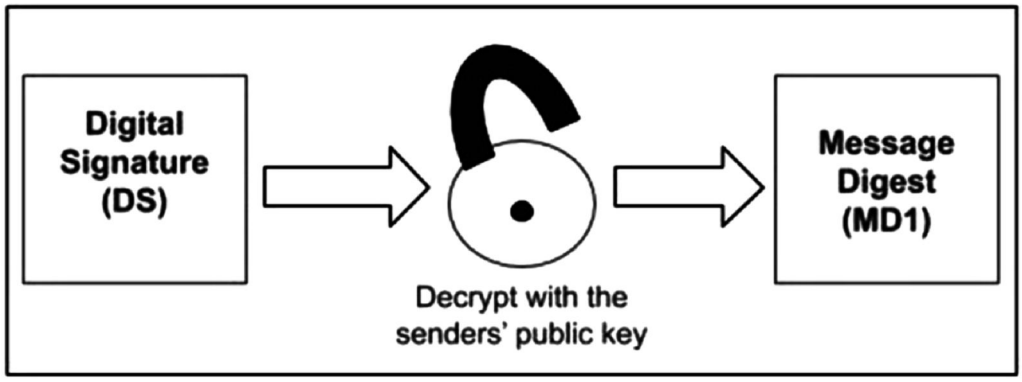
When B receives the Original Message(M) and the Digital Signature(DS) from A, it first uses the same message-digest algorithm as was used by A and calculates its own Message Digest (MD2) for M.



Receiver calculates its own message digest

Step-5 :

Now B uses A's public key to decrypt the digital signature because it was encrypted by A's private key. The result of this process is the original Message Digest (MD1) which was calculated by A.

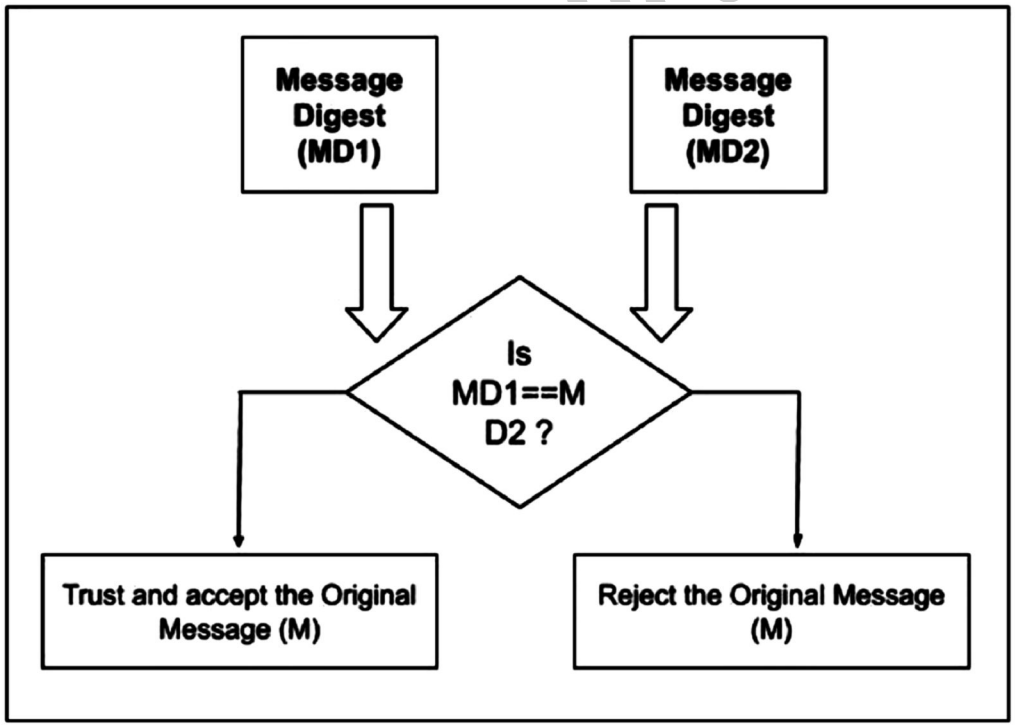


Receiver retrieves sender's message digest

Step-6 :

If MD1 = = MD2, the following facts are established as follows.

- B accepts the original message M as the correct, unaltered message from A.
- It also ensures that the message came from A and not someone posing as A.



Digital signature verification

The message digest (MD1) was encrypted using A's private key to produce a digital signature. Therefore, the digital signature can be decrypted using A's public key (due to asymmetric form of RSA). If the receiver B is able to decrypt the digital signature using A's public key, it means that the message is received from A itself and now A cannot deny that he/she has not sent the message.

It also proves that the original message did not tamper because when the receiver B tried to find its own message digest MD2, it matched with that of A's MD1.

Suppose a malicious user tries to access the original message and perform some alteration. Now he/she will calculate a new message digest over the altered message. It might concern you with data integrity and confidentiality but here's the catch. The attacker will have to sign the altered message using A's private key in order to pose as A for the receiver B. However, an attacker cannot sign the message with A's private key because it is known to A only. Hence, the RSA signature is quite strong, secure, and reliable.

Attacks on RSA Signature :

There are some attacks that can be attempted by attackers on RSA digital signatures. A few of them are given below as follows.

1. Chosen-message Attack

In the chosen-message attack, the attacker creates two different messages, M1 and M2, and somehow manages to persuade the genuine user to sign both the messages using RSA digital-signature scheme. Let's consider message M1 and message M2. so, the attacker computes a new message M = M1 x M2 and then claims that the genuine user has signed message M.

2. Key-only Attack

In this attack, the Assumption is that attacker has access to the genuine user public key and tries to get a message and digital signature. Only The attacker then tries to create another message MM such that the same signature S looks to be valid on MM. However, it is not an easy attack to launch since the mathematical complexity beyond this is quite high.

3. Known-message Attack

In a known-message attack, the attacker tries to use a feature of RSA whereby two different messages having two different signatures can be combined so that their signatures also combine. To take an example, let us say that we have two different messages M1 and M2 with respective digital signatures as S1 and S2. Then if M = (M1 x M2) mod n, mathematically S = (S1 ¥ S2) mod n. Hence, the attacker can compute M = (M1 x M2) mod n and then S = (S1 x S2) mod n to forge a signature.

3.5.2 DSA Signatures

Q11. Explain DSA algorithm.

Ans : **(Imp.)**

The DSA is based on the difficulty of computing discrete logarithms and is based on schemes originally presented by ElGamal [ELGA85] and Schnorr [SCHN91].

Summarizes the algorithm. There are three parameters that are public and can be common to a group of users. A 160-bit prime number q is chosen.

Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$

for $512 \leq L \leq 1024$ and L a multiple of 64;

i.e., bit length of between 512 and 1024 bits

in increments of 64 bits

q prime divisor of (p –1). where $2^{159} < q < 2^{160}$;

i.e.. bit length of 160 bits

g = $h^{ip-1)q}$ modp,

where h is any integer with 1 < h < (p – 1)

such that $h^{(p-1)/q}$ mod p > 1

User's Private Key

x random or pseudorandom integer with 0 < x < q

User's Public Key

y = g^x mod p

User's Per-Message Secret Number

k = random or pseudorandom integer with 0 < k < q

Signing

r = (g^k mod p) mod q

s = F (H(M) + .vr)] mod q

Signature = (r, s)

Verifying

w = $(s')^{-1}$ mod 1

v_i = [H(M')w] mod q

u_2 = (r')w mod q

v = [($g^{"1}$ $v^{"2}$) mod p) mod q

TEST: v = r

M = message to be signed

H(M) = hash of M using SHA-1

M', r',s' = received versions of M, r, s

Fig.: The Digital Signature Algorithm (DSA)

Next, a prime number p is selected with a length between 512 and 1024 bits such that q-divides (p – 1). Finally, g is chosen to be of the form $h^{(p-1)q}$ mod p where h is an integer between 1 and (p – 1) with the restriction that g must be greater than l.

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

To create a signature, a user calculates two quantities, r and s, that are functions of the public key components (p, q, g), the user's private key (x), the hash code of the message, H(M), and an additional integer k that should be generated randomly or pseudorandomly and be unique for each signing.

At the receiving end, verification is performed using the formulas shown in fig. The receiver generates a quantity v that is a function of the public key components, the sender's public key, and the hash code of the incoming message. If this quantity matches the r component of the signature, then the signature is validated.

Fig. depicts the functions of signing and verifying.

The structure of the algorithm, as revealed in Fig. is quite interesting. Note that the test at the end is on the value r, which does not depend on the

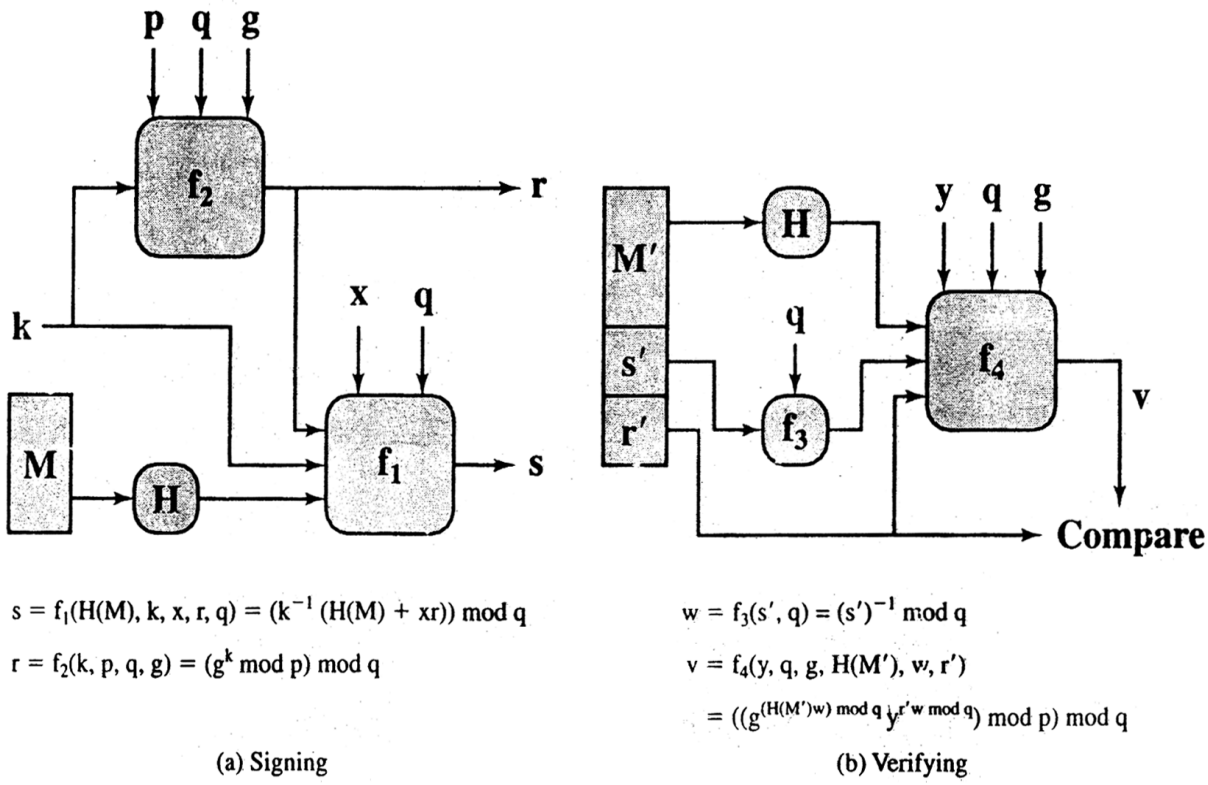


Fig.: DSS Signing and Verifying

message at all. Instead, r is a function of k and the three global public-key components. The multiplicative inverse of k (mod q) is passed to a function that also has as inputs the message hash code and the user's private key. The structure of this function is such that the receiver can recover r using the incoming message and signature, the public key of the user, and the global public key. It is certainly not obvious from fig or fig. that such a scheme would work. A proof is provided at this book's Web site.

Given the difficulty of taking discrete logarithms, it is infeasible for an opponent to recover k from r or to recover x from s.

Another point worth noting is that the only computationally demanding task in signature generation is the exponential calculation g^k mod p. Because this value does not depend on the message to be signed, it can be computed ahead of time. Indeed, a user could precalculate a number of values of r to be used to sign documents as needed. The only other somewhat demanding task is the determination of a multiplicative inverse. k^{-1} Again, a number of these values can be precalculated.

3.6 BIOMETRIC AUTHENTICATION

Q12. Define Biometric Authentication. Explain its types.

Ans : **(Imp.)**

Meaning

Biometric authentication is a type of system that relies on the unique biological characteristics of individuals to verify identity for secure access to electronic systems.

Biometric verification is considered a subset of biometric authentication. The biometric technologies involved are based on the ways in which individuals can be uniquely identified through one or more distinguishing biological traits, such as fingerprints, hand geometry, earlobe geometry, retina and iris patterns, voice waves, keystroke dynamics, DNA and signatures. Biometric authentication is the application of that proof of identity as part of a process validating a user for access to a system. Biometric technologies are used to secure a wide range of electronic communications, including enterprise security, online commerce and banking even just logging in to a computer or smartphone.

Biometric authentication systems compare the current biometric data capture to stored, confirmed authentic data in a database. If both samples of the biometric data match, authentication is confirmed and access is granted. The process is sometimes part of amultifactor authentication system. For example, a smartphone user might log on with his personal identification number (PIN) and then provide an iris scan to complete the authentication process.

Types

(i) Retina scan

Iris recognition is used to identify individuals based on unique patterns within the ring- shaped region surrounding the pupil of the eye.

(ii) Fingerscanning

Fingerscanning, the digital version of the ink-and-paper fingerprinting process, works with details in the pattern of raised areas and branches in a human finger image.

(iii) Finger vein ID

Finger vein ID is based on the unique vascular pattern in an individual's finger.

(iv) Facial recognition

Facial recognition systems work with numeric codes called faceprints, which identify 80 nodal points on a human face.

(v) Voice identification systems

Voice identification systems rely on characteristics created by the shape of the speaker's mouth and throat, rather than more variable conditions.

Once seen mostly in spy movies (where it might be used to protect access to a top-secret military lab, for example), biometric authentication is becoming relatively commonplace. In addition to the security provided by hard-to-fake individual biological traits, the acceptance of biometric verification has also been driven by convenience: One can't easily forget or lose ones biometrics.

Q13. Explain the evolution of biometric verification.

Ans : **(Imp.)**

The oldest known use of biometric verification is fingerprinting. Thumbprints made on clay seals were used as a means of unique identification as far back as ancient China.

Modern biometric verification has become almost instantaneous, and is increasingly accurate with the advent of computerized databases and the digitization of analog data.

The market for biometrics products is still too fractured to name specific top providers. The physical characteristics of the biometrics products available today vary from the mundane, such as fingerprinting, to the esoteric, like typing speeds and electrophysiological signals.

Until recently, biometrics was typically used at a physical security level – protecting facilities at military bases or impenetrable bank vaults, for example. But, because single-factor authentication methods are easy to break, companies have started looking to two-factor solutions, like biometrics.

Q14. Explain the barriers biometric authentication.

Ans :

However, the following five fundamental barriers may limit the growth of biometric authentication:

1. Biometrics can be complicated and costly to deploy. All biometric deployments require installation of their own hardware and application servers.
2. The market is still fractured. Should you buy a fingerprint reader, a voice recognition system or an iris scanner? Since each product differs greatly in its approach and installation, it is difficult to compare them during a typical company bid process.
3. Biometric data is like any other data. It sits on servers, which are bait forhackers if not properly hardened and secured. Therefore, when reviewing any biometric product, make sure it transmits data securely, meaningencrypted, from the biometric reader back to the authenticating server. And, make sure the authenticating server has been hardened, patched and protected.

4. Biometric readers are prone to errors. Fingerprints can smudge, faces and voices can be changed and all of them can be misread, blocking a legitimate user, or permitting access to an unauthorized or malicious user.

5. Difficulties with user acceptance. Properly trained employees may be willing to use biometrics devices, but customers, like those logging on to your Web site, may be more reluctant to use – or worse, forced to purchase – a device that's difficult to use or makes doing business, such as banking, on your site, a hassle instead of a convenience. And both your employees and customers may be squeamish about exposing their eyes to devices like iris scanners, even if they appear harmless.

Despite these issues, biometrics is slowly gaining acceptance for two-factor authentication purposes. The products are getting better, lighter and easier to use. Error rates are going down, and fingerprint readers installed on tokens and laptops are getting smaller and less intrusive. And, like the rest of the security product industry, vendors will eventually merge and consolidate, uniting a fractured market, which will make it easier to choose a product that suits your business needs.

UNIT IV

PKI Interface

Digital Certificates, Certifying Authorities, POP Key Interface, System Security using Firewalls and VPN's.

Smart Cards

Application Security using Smart Cards, Zero Knowledge Protocols and their use in Smart Cards, Attacks on Smart Cards.

4.1 PKI INTERFACE

Q1. What do you mean by PKI interface.

(OR)

Explain about PKI interface.

Ans : **(Imp.)**

A public key infrastructure (PKI) is a set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption. The purpose of a PKI is to facilitate the secure electronic transfer of information for a range of network activities such as e-commerce, internet banking and confidential email. It is required for activities where simple passwords are an inadequate authentication method and more rigorous proof is required to confirm the identity of the parties involved in the communication and to validate the information being transferred.

In cryptography, a PKI is an arrangement that binds public keys with respective identities of entities (like people and organizations). The binding is established through a process of registration and issuance of certificates at and by a certificate authority (CA). Depending on the assurance level of the binding, this may be carried out by an automated process or under human supervision. When done over a network, this requires using a secure certificate enrollment or certificate management protocol such as CMP.

PKI provides assurance of public key. It provides the identification of public keys and their distribution. An anatomy of PKI comprises of the following components.

- Public Key Certificate, commonly referred to as 'digital certificate'.
- Certification Authority.
- Registration Authority.
- Certificate Management System.

4.1.1 Digital Certificate, Certifying authority

Q2. Explain the concept of Digital Certificate and CA.

(OR)

Explain about Certifying Authority.

Ans : **(Imp.)**

For analogy, a certificate can be considered as the ID card issued to the person. People use ID cards such as a driver's license, passport to prove their identity. A digital certificate does the same basic thing in the electronic world, but with one difference.

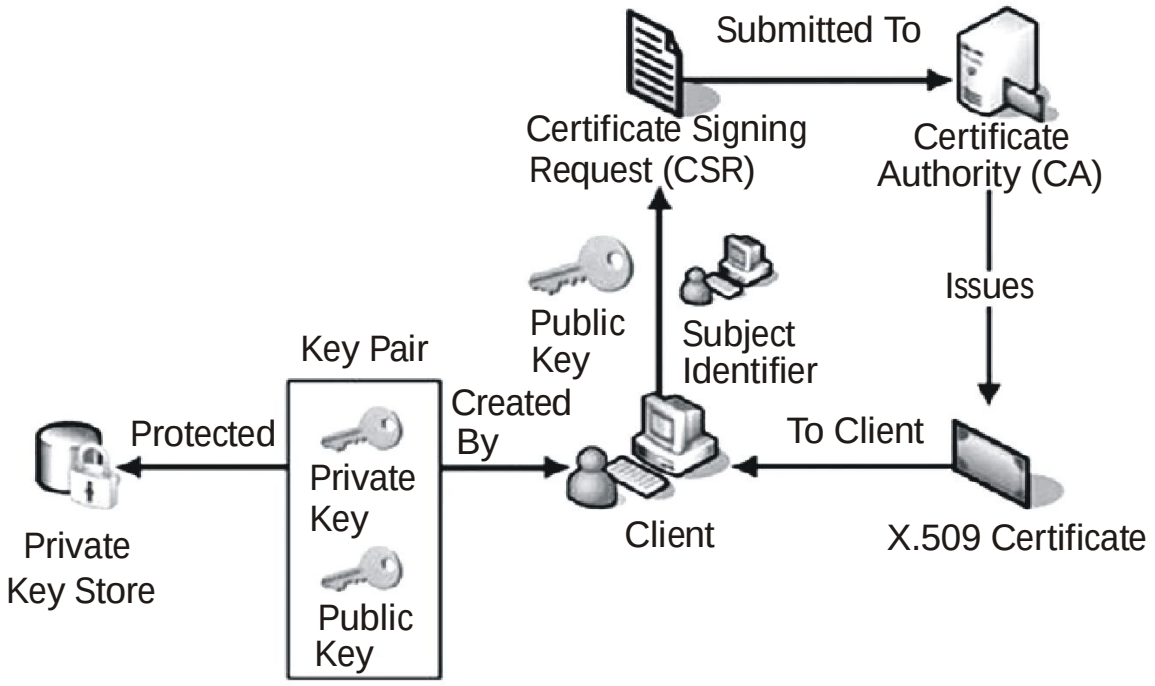
Digital Certificates are not only issued to people but they can be issued to computers, software packages or anything else that need to prove the identity in the electronic world.

- Digital certificates are based on the ITU standard X.509 which defines a standard certificate format for public key certificates and certification validation. Hence digital certificates are sometimes also referred to as X.509 certificates.

  Public key pertaining to the user client is stored in digital certificates by The Certification Authority (CA) along with other relevant information such as client information, expiration date, usage, issuer etc.

- CA digitally signs this entire information and includes digital signature in the certificate.
- Anyone who needs the assurance about the public key and associated information of client, he carries out the signature validation process using CA's public key. Successful validation assures that the public key given in the certificate belongs to the person whose details are given in the certificate.

The process of obtaining Digital Certificate by a person/entity is depicted in the following illustration.



As shown in the illustration, the CA accepts the application from a client to certify his public key. The CA, after duly verifying identity of client, issues a digital certificate to that client.

Certifying Authority (CA)

The CA issues certificate to a client and assist other users to verify the certificate. The CA takes responsibility for identifying correctly the identity of the client asking for a certificate to be issued, and ensures that the information contained within the certificate is correct and digitally signs it.

Key Functions of CA

The key functions of a CA are as follows -

- **Generating key pairs**

  The CA may generate a key pair independently or jointly with the client.

- **Issuing digital certificates**

  The CA could be thought of as the PKI equivalent of a passport agency " the CA issues a certificate after client provides the credentials to confirm his identity. The CA then signs the certificate to prevent modification of the details contained in the certificate.

- **Publishing Certificates**

  The CA need to publish certificates so that users can find them. There are two ways of achieving this. One is to publish certificates in the equivalent of an electronic telephone directory. The other is to send your certificate out to those people you think might need it by one means or another.

- **Verifying Certificates**

  The CA makes its public key available in environment to assist verification of his signature on clients' digital certificate.

- **Revocation of Certificates**

  At times, CA revokes the certificate issued due to some reason such as compromise of private key by user or loss of trust in the client. After revocation, CA maintains the list of all revoked certificate that is available to the environment.

  Classes of Certificates

  There are four typical classes of certificate -

  - **Class 1 :** These certificates can be easily acquired by supplying an email address.
  - **Class 2 :** These certificates require additional personal information to be supplied.
  - **Class 3 :** These certificates can only be purchased after checks have been made about the requestor's identity.
  - **Class 4 :** They may be used by governments and financial organizations needing very high levels of trust.

Registration Authority (RA)

CA may use a third-party Registration Authority (RA) to perform the necessary checks on the person or company requesting the certificate to confirm their identity. The RA may appear to the client as a CA, but they do not actually sign the certificate that is issued.

Certificate Management System (CMS)

It is the management system through which certificates are published, temporarily or permanently suspended, renewed, or revoked. Certificate management systems do not normally delete certificates because it may be necessary to prove their status at a point in time, perhaps for legal reasons. A CA along with associated RA runs certificate management systems to be able to track their responsibilities and liabilities.

Private Key Tokens

While the public key of a client is stored on the certificate, the associated secret private key can be stored on the key owner's computer. This method is generally not adopted. If an attacker gains access to the computer, he can easily gain access to private key. For this reason, a private key is stored on secure removable storage token access to which is protected through a password.

Different vendors often use different and sometimes proprietary storage formats for storing keys. For example, Entrust uses the proprietary. epf format, while Verisign, Global Sign, and Baltimore use the standard .p12 format.

Hierarchy of CA

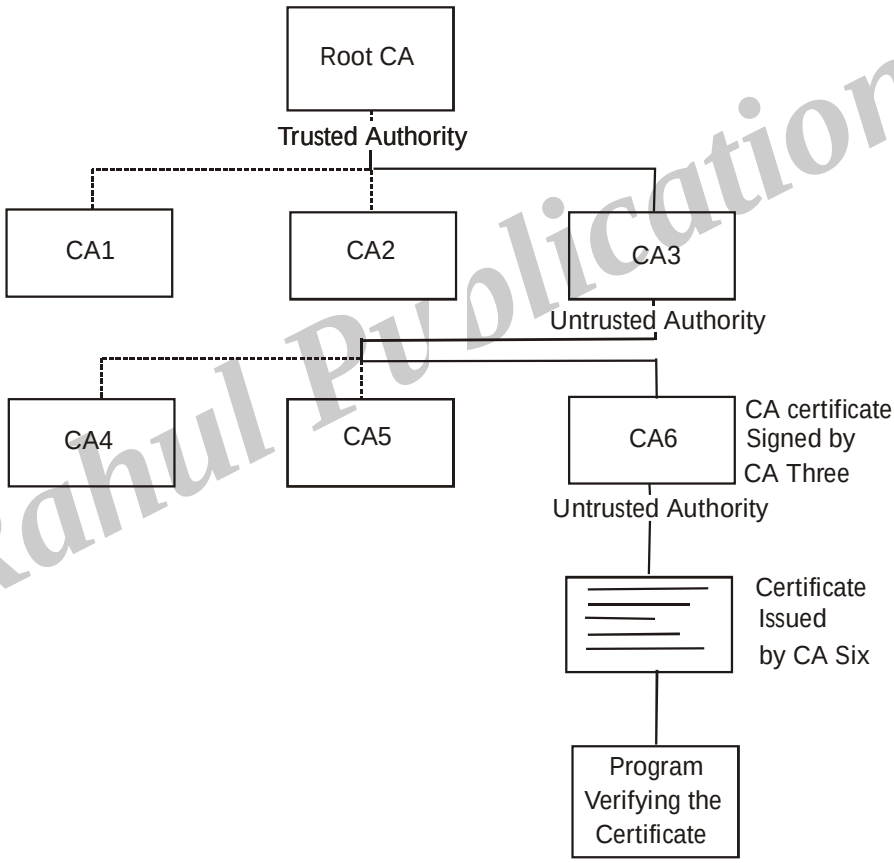
With vast networks and requirements of global communications, it is practically not feasible to have only one trusted CA from whom all users obtain their certificates. Secondly, availability of only one CA may lead to difficulties if CA is compromised.

In such case, the hierarchical certification model is of interest since it allows public key certificates to be used in environments where two communicating parties do not have trust relationships with the same CA.

- The root CA is at the top of the CA hierarchy and the root CA's certificate is a self-signed certificate.
- The CAs, which are directly subordinate to the root CA (For example, CA1 and CA2) have CA certificates that are signed by the root CA.
- The CAs under the subordinate CAs in the hierarchy (For example, CA5 and CA6) have their CA certificates signed by the higher-level subordinate CAs.

Certificate authority (CA) hierarchies are reflected in certificate chains. A certificate chain traces a path of certificates from a branch in the hierarchy to the root of the hierarchy.

The following illustration shows a CA hierarchy with a certificate chain leading from an entity certificate through two subordinate CA certificates (CA6 and CA3) to the CA certificate for the root CA.



Verifying a certificate chain is the process of ensuring that a specific certificate chain is valid, correctly signed, and trustworthy. The following procedure verifies a certificate chain, beginning with the certificate that is presented for authentication-

- A client whose authenticity is being verified supplies his certificate, generally along with the chain of certificates up to Root CA.
- Verifier takes the certificate and validates by using public key of issuer. The issuer's public key is found in the issuer's certificate which is in the chain next to client's certificate.

- Now if the higher CA who has signed the issuer's certificate, is trusted by the verifier, verification is successful and stops here.
- Else, the issuer's certificate is verified in a similar manner as done for client in above steps. This process continues till either trusted CA is found in between or else it continues till Root CA.

Q3. Describe the best practices for PKI Management.

(OR)

Explain the best practices for PKI Management.

Ans : **(Imp.)**

Establishing and managing an ideal PKI system would involve an impeccably managed infrastructure that included certificates and keys, CAs, HSMs, associated DevOps, ITSM, and IAM tools, and a lot more. The management of each of those systems is a vast topic that we will be covering in later articles. For now, here are some high-level considerations prior to (or during, or even after) a PKI implementation.

- **Maintain a certificate inventory**

  Ensure that every certificate – currently in use, discarded, or revoke – is tracked in a centralized inventory system, where every certificate is mapped to the endpoint it is (or was) installed on. This simplifies renewals and maintenance, and removes the risk associated with rogue or unused certificates that can be exploited.

- **Protect private keys**

  Too often, private keys are stored on and shared in the form of text documents, or password-protected files. This is a bad practice, since a compromised private key to a root or intermediate CA could sabotage an entire portion of a network. Ensure that you use HSMs that meet compliance requirements (FIPS 140-2) to store keys, and secure vaults to store passwords. Ideally, a private key should undergo automated rotation from within the HSM, and should never be manually handled by a human.

- **Use certificates issued by trusted CAs**

  For external use, ensure that you purchase certificates that are issued by globally trusted Certificate Authorities, as opposed to self-signed certificates. These self-signed certificates are usually issued for internal use, and are signed by your own organization – they are common in testing scenarios. However, using such certificates on external-facing applications or endpoints results in that endpoint not being trusted, and also highly vulnerable to misuse.

- **Rotate certificates, keys, and SSH keys**

  Public Key Infrastructure certificates are increasingly becoming more short-lived by the day. However, don't restrict yourself to the limits set by browsers. Certificates with shorter validities are always more secure (new technology like DevOps, IoT, and cloud applications are already on board the short-lived certificate train), and naturally, the keys have to be rotated when the certificates are renewed, too. SSH keys must also be rotated on a frequent basis, since they're password-based and can be cracked easily if left stale for a long period of time.

- **Establish policy**

  Create and enforce PKI management policies with regard to role-based access to crypto-assets, certificate renewal durations, and PKI audit/audit trails. By creating transparency and clearly defined rules, the chances of mismanagement-induced vulnerabilities are lowered, and errors, if any, can easily be tracked and remediated.

- **Practice crypto-agility**

  Imbue PKI with a system of control that allows for accelerated manipulation of its constituent systems. This includes the ability to quickly rotate certificates, expedite the enrollment/ renewal/revocation process with CAs, and rapidly switch out outdated algorithms and protocols with new ones. The logic behind crypto-agility is that an administrator should be able to quickly remediate vulnerabilities in a crypto-system without disrupting the

network environment as a whole. This is an ability that comes in handy when existing methods are phased out in favor of new ones. Some examples include the switch from SHA-1 to SHA-2, and the deprecation of TLS 1.0 and 1.1, in favor of 1.2 and 1.3.

4.2 Pop Key Interface

Q4. Explain about Pop Key Interface.

Ans :

Point of presence (POP) is the point at which two or more different networks or communication devices build a connection with each other. POP mainly refers to an access point, location or facility that connects to and helps other devices establish a connection with the Internet.

POP is primarily the infrastructure that allows remote users connect to connect to the Internet. A POP is generally present at an Internet service provider (ISP) or the telecommunication service provider. It can consist of a router, switches, servers and other data communication devices. An ISP or telecom provider might maintain more than one POP at different locations, with each catering to a distinct user base. Moreover, POP also supports the conversion of analog to digital data and vice versa to complement different data communication technologies and receiving devices.

4.3 System Security Using Firewalls

Q5. Define Firewall. Listout various firewall design principles.

(OR)

What is firewall.

(OR)

Explain the principles of firewalls.

Ans :

Firewalls can be an effective means of protecting a local system or network of system from network-based security threats while at the same time affording access to the out side world via wide area networks and the Internet.

Principles

Information systems in corporations, government agencies, and other organization have undergone a steady evolution:

- Centralized data processing system, with a central mainframe supporting a number of directly connected terminals.
- Local area networks (LANs) interconnecting PCs and terminals to each other and the mainframe.
- Premises network, consisting of a number of LANs, interconnecting PCs, servers, and perhaps a mainframe or two.
- Enterprise-wide network, consisting of multiple, geographically distributed premises networks interconnected by a private wide area network (WAN)
- Internet connectivity, in which the various premises networks all hook into the Internet and may or may not also be connected by a private WAN.

Q6. Listant various goals for a firewall.

Ans :

The following are the goals for a firewall:

1. All traffic from inside to outside, and vice versa, must pass through the firewall. This is achieved by physically blocking all access to the local network except via the firewall. Various configurations are possible, as explained later in this section.
2. Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies, as explained later in this section.
3. The firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system.

   The lists four general techniques that firewalls use to control access and enforce the site's

security policy. Originally, firewalls focused primarily on service control, but they have since evolved to provide all four:

- **Service control**

  Determines the types of Internet services that can be accessed, inbound or outbound. The firewall may filter traffic on the basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as a Web or mail service.

- **Direction control**

  Determines the direction in which particular service requests may be initiated and allowed to flow through the firewall.

- **User control**

  Controls access to a service according to which user is attempting to access it. This feature is typically applied to users inside the firewall perimeter (local users). It may also be applied to incoming traffic from external users; the latter requires some form of secure authentication technology, such as is provided in IPSec (Chapter 16).

- **Behavior control**

  Controls how particular services are used. For example, the firewall may filter e-mail to eliminate spam, or it may enable external access to only a portion of the information on a local Web sever.

Q7. Explain different types of firewalls.

Ans : **(Imp.)**

There are three common types of firewalls: (i) Packet filters (ii) Application - level gateways and (iii) circuit-level gateways.

(i) Packet-Filtering Router

A packet-filtering router applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions (from and to the internal network). Filtering rules are based on information contained in a net-work packet:

- **Source IP address**

  The IP address of the system that originated the IP packet (e.g., 192.178.1.1)

- **Destination IP address**

  The IP address of the system the IP packet is trying to reach (e.g., 192.168.1.2)

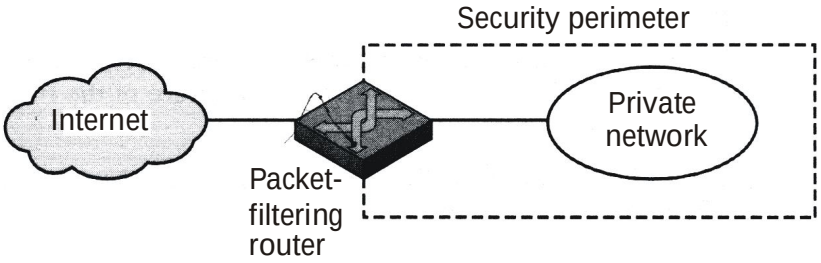
- **Source and destination transport-level address**

  The transport level (e.g.. TCP or UDP) port number, which defines applications such as SNMP or TELNET.

- **IP protocol field:** Defines the transport protocol

- **Interface**

  For a router with three or more ports, which interface of the router the packet came from or which interface of the router the packet is destined for



(a) Packet-filtering router

The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken. 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 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 more likely to see the firewall as a hindrance. The default forward policy increases ease of use for end users but provides reduced security; the security administrator must, in essence, react to each new security threat as it becomes known.

Table 20.1, from [BELL94b], gives some examples of packet-filtering rule sets. In each set, the rules are applied top to bottom. The "*" in a field is a wildcard designator that matches everything. We assume that the default = discard policy is in force.

A. Inbound mail is allowed (port 25 is for SMTP incoming), but only to a gateway host. However, packets from a particular external host, SPIGOT, are blocked because that host has a history of sending massive files in e-mail messages.

B. This is an explicit statement of the default policy. All rule sets include this rule implicitly as the last rule.

C. This rule set is intended to specify that any inside host can send mail to the outside. A TCP packet with a destination port of 25 is routed to the SMTP server on the destination machine. The problem with this rule is that the use of port 25 for SMTP receipt is only a default; an outside machine could be configured to have some other application linked to port 25. As this rule is written, an attacker could gain access to internal machines by sending packets with a TCP source port number of 25.

D. This rule set achieves the intended result that was not achieved in C. The rules take advantage of a feature of TCP connections. Once a connection is set up, the ACK flag of a TCP segment is set to acknowledge segments sent from the other side. Thus, this rule set states that it allows IP packets where the source IP address is one of a list of designated internal hosts and the destination TCP port number is 25. It also allows incoming packets with a source port number of 25 that include the ACK flag in the TCP segment. Note that we explicitly designate source and destination systems to define these rules explicitly.

A

action	ourhost	port	theirhost	port	comment
block	*	*	SPIGOT	*	wedon't trust these people
allow	OUR-GW	25	*	*	connection to our SMTP port

B

action	ourhost	port	theirhost	port	comment
block	*	*	*	*	default

C

action	ourhost	port	theirhost	port	comment
allow	*	*	*	25	connection to their SMTP port

D

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	25		our packets to their SMTP port
allow	*	25	*	*	ACK	their replies

E

action	src	port	dest	port	flags	comment
allow	{our hosts}	*	*	*		our outgoing calls
allow	*	*	*	*	ACK	replies to our calls
allow	*	*	*	>1024		traffic to nonservers

Fig.: Packet - Filtering Examples

Attacks

Some of the attacks that can be made on packet-filtering routers and the appropriate countermeasures are the following:

- **IP address spoofing**

  The intruder transmits packets from the outside with a source IP address field containing an address of an internal host. The attacker hopes that the use of a spoofed address will allow penetration of systems that employ simple source address security, in which packets from specific trusted internal hosts are accepted. The countermeasure is to discard packets with an inside source address if the packet arrives on an external interface.

- **Source routing attacks**

  The source station specifies the route that a packet should take as it crosses the Internet, in the hopes that this will bypass security measures that do not analyze the source routing information. The countermeasure is to discard all packets that use this option.

- **Tiny fragment attacks**

  The intruder uses the IP fragmentation option to create extremely small fragments and force the TCP header information into a separate packet fragment. This attack is designed to circumvent filtering rules that depend on TCP header information. Typically, a packet filter will make a filtering decision on the first fragment of a packet. All subsequent fragments of that packet are filtered out solely on the basis that they are part of the packet whose first fragment was rejected. The attacker hopes that the filtering router examines only the first fragment and that the remaining fragments are passed through. A tiny fragment attack can be defeated by enforcing a rule that the first fragment of a packet must contain a predefined minimum amount of the transport header. If the first fragment is rejected, the filter can remember the packet and discard all subsequent fragments.

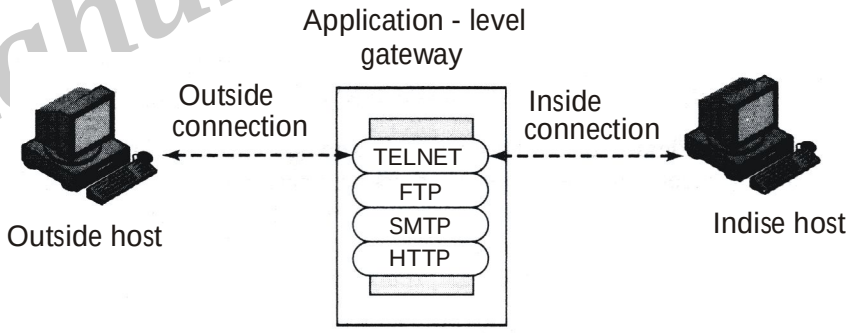
(ii) Application-Level Gateway

An application - level gateway, also called a proxy server, acts as a relay of application- level traffic. The user contacts

Source Address	Source Port	Destination	Destination Port	Connection
192.168.1.100	1030	210.9.88.29	80	Established
192.168.1.102	1031	216.32.42.123	80	Established
192.168.1.101	1033	173.66.32.122	25	Established
192.168.1.106	1035	177.231.32.12	79	Established
223.43.21.231	1990	192.168.1.6	80	Established
219.22.123.32	2112	192.168.1.6	80	Established
210.99.212.18	3321	192.168.1.6	80	Established
24.102.32.23	1025	192.168.1.6	80	Established
223.212.212	1046	192.168.1.6	80	Established

Table: Example Stateful Firewall Connection State Table [WACK 02]

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. If the gateway does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall. Further, the gateway can be configured to support only specific features of an application that the network administrator considers acceptable while denying all other features.



(b) Application - level gateway

Application-level gateways tend to be more secure than packet filters. Rather than trying to deal with the numerous possible combinations that are to be allowed and forbidden at the TCP and IP level, the application-level gateway need only scrutinize a few allowable applications. In addition, it is easy to log and audit all incoming traffic at the application level.

A prime disadvantage of this type of gateway is the additional processing overhead on each connection. In effect, there are two spliced connections between the end users, with the gateway at the splice point, and the gateway must examine and forward all traffic in both directions.

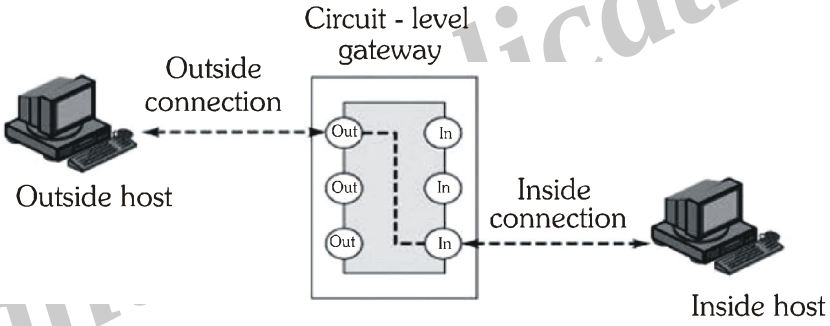
(iii) Circuit-Level Gateway

A third type of firewall is the circuit-level gateway. This can be a stand-alone system or it can be a specialized 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 outside 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. In this configuration, the gateway can incur the processing overhead of examining incoming application data for forbidden functions but does not incur that overhead on outgoing data.

An example of a circuit-level gateway implementation is the SOCKS package [KOBL92]; version 5 of SOCKS is defined in RFC 1928. The RFC defines SOCKS in the following fashion:

The protocol described here is designed to provide a framework for client-server applications in both the TCP and UDP domains to conveniently and securely use the services of a network firewall. The protocol is conceptually a "shim-layer" between the application layer and the transport layer, and as such does not provide network-layer gateway services, such as forwarding of ICMP messages.



(c) Circuit - level gateway

SOCKS consists of the following components:

- The SOCKS server, which runs on a UNIX-based firewall.
- The SOCKS client library, which runs on internal hosts protected by the firewall.
- SOCKS-ified versions of several standard client programs such as FTP and TELNET. The implementation of the SOCKS protocol typically involves the recompilation or relinking of TCP-based client applications to use the appropriate encapsulation routines in the SOCKS library.

When a TCP-based client wishes to establish a connection to an object that is reachable only via a firewall (such determination is left up to the implementation), it must open a TCP connection to the appropriate SOCKS port on the SOCKS server system. The SOCKS service is located on TCP port 1080. If the connection request succeeds, the client enters a negotiation for the authentication method to be used, authenticates with the chosen method, and then sends a relay request. The SOCKS server evaluates the request and either establishes the appropriate connection or denies it. UDP exchanges are handled in a similar fashion. In essence, a TCP connection is opened to authenticate a user to send and receive UDP segments, and the UDP segments are forwarded as long as the TCP connection is open.

Bastion Host A bastion host is a system identified by the firewall administrator as a critical strong point in the network's security. Typically, the bastion host serves as a platform for an application-level or circuit-level gateway. Common characteristics of a bastion host include the following:

- The bastion host hardware platform executes a secure version of its operating system, making it a trusted system.
- Only the services that the network administrator considers essential are installed on the bastion host. These include proxy applications such as Telnet, DNS, FTP, SMTP, and user authentication.
- The bastion host may require additional authentication before a user is allowed access to the proxy services. In addition, each proxy service may require its own authentication before granting user access.
- Each proxy is configured to support only a subset of the standard application's command set.
- Each proxy is configured to allow access only to specific host systems. This means that the limited command/feature set may be applied only to a subset of systems on the protected network.
- Each proxy maintains detailed audit information by logging all traffic, each connection, and the duration of each connection. The audit log is an essential tool for discovering and terminating intruder attacks.
- Each proxy module is a very small software package specifically designed for net-work security. Because of its relative simplicity, it is easier to check such modules for security flaws. For example, a typical UNIX mail application may contain over 20,000 lines of code, while a mail proxy may contain fewer than 1000.
- Each proxy is independent of other proxies on the bastion host. If there is a problem with the operation of any proxy, or if a future vulnerability is discovered, it can be uninstalled without affecting the operation of the other proxy applications. Also, if the user population requires support for a new service, the network administrator can easily install the required proxy on the bastion host.
- A proxy generally performs no disk access other than to read its initial configuration file. This makes it difficult for an intruder to install Trojan horse sniffers or other dangerous files on the bastion host.
- Each proxy runs as a nonprivileged user in a private and secured directory on the bastion host.

Q8. Explain the concept of firewall configuration.

Ans : **(Imp.)**

In addition to the use of a simple configuration consisting of a single system, such as a single packet-filtering router or a single gateway more complex con-figurations are possible and indeed more common.

(i) Screened host firewall, single - homed bastion

In the screened host firewall, single-homed bastion configuration , the firewall consists of two systems: a packet-filtering router and a bastion host. Typically, the router is configured so that

1. For traffic from the Internet, only IP packets destined for the bastion host are allowed in.
2. For traffic from the internal network, only IP packets from the bastion host are allowed out.

The bastion host performs authentication and proxy functions. This configuration has greater security than simply a packet-filtering router or an application-level gateway alone, for two reasons. First, this configuration implements both packet- level and application-level filtering, allowing for considerable flexibility in defining security policy. Second, an intruder must generally penetrate two separate systems before the security of the internal network is com-promised.

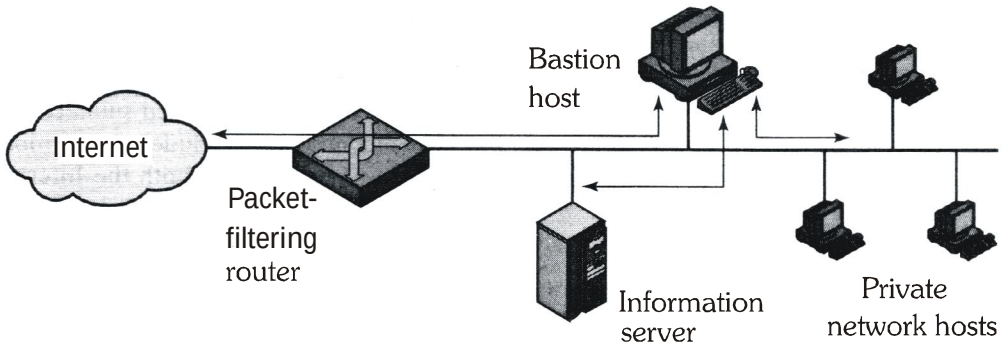


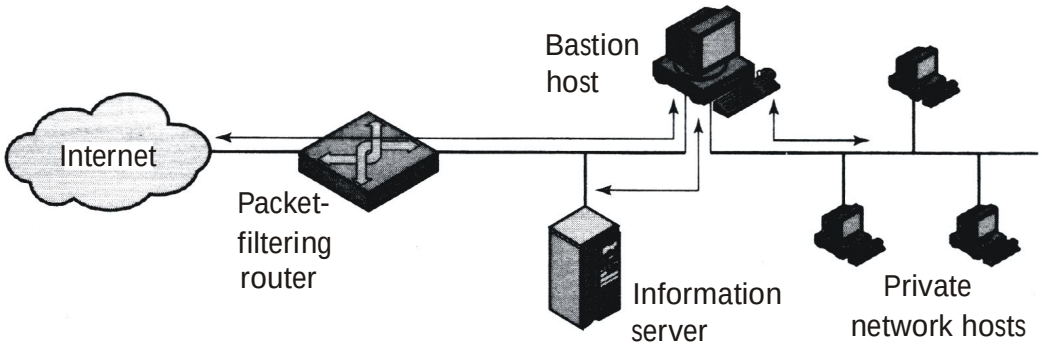
Fig.: (a) Screened firewall system (single-homed bastion host)

This configuration also affords flexibility in providing direct Internet access. For example, the internal network may include a public information server, such as a Web server, for which a high level of security is not required. In that case, the router can be configured to allow direct traffic between the information server and the Internet.

In the single-homed configuration just described, if the packet-filtering router is completely compromised, traffic could flow directly through the router between the Internet and other hosts on the private network.

(ii) Screen Host Firewall

The screened host firewall, dual-homed bastion configuration physically prevents such a security breach . The advantages of dual layers of security that were present in the previous configuration are present here as well. Again, an information server or other hosts can be allowed direct communication with the router if this is in accord with the security policy.



(b) Screened host firewall system (dual - homed bastion host)

(iii) Screened subnet firewall configuration

The screened subnet firewall configuration is the most secure of those we have considered. In this configuration, two packet-filtering routers are used, one between the bastion host and the Internet and one between the bastion host and the internal network. This configuration creates an isolated subnetwork, which may consist of simply the bastion host but may also include one or more information servers and modems for dial-in capability. Typically, both the Internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked. This configuration offers several advantages:

- There are now three levels of defense to thwart intruders.
- The outside router advertises only the existence of the screened subnet to the Internet; therefore, the internal network is invisible to the Internet.

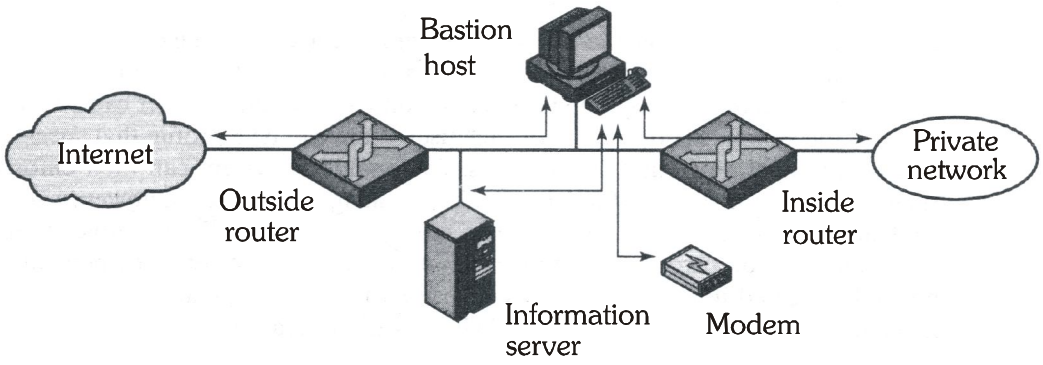


Fig.: Screened - subnet firewall system

- Similarly, the inside router advertises only the existence of the screened subnet to the internal network; therefore, the systems on the inside network cannot construct direct routes to the Internet.

4.3.1 VPN'S

Q9. What is VPN. State its benefits.

(OR)

Describe the benefits of VPN.

Ans :

Meaning

VPN stands for "Virtual Private Network" and describes the opportunity to establish a protected network connection when using public networks. VPNs encrypt your internet traffic and disguise your online identity. This makes it more difficult for third parties to track your activities online and steal data. The encryption takes place in real time.

Working

A VPN hides your IP address by letting the network redirect it through a specially configured remote server run by a VPN host. This means that if you surf online with a VPN, the VPN server becomes the source of your data. This means your Internet Service Provider (ISP) and other third parties cannot see which websites you visit or what data you send and receive online. A VPN works like a filter that turns all your data into "gibberish". Even if someone were to get their hands on your data, it would be useless.

Benefits

A VPN connection disguises your data traffic online and protects it from external access. Unencrypted data can be viewed by anyone who has network access and wants to see it. With a VPN, hackers and cyber criminals can't decipher this data.

(i) Secure encryption

To read the data, you need an encryption key. Without one, it would take millions of years for a computer to decipher the code in the event of a brute force attack . With the help of a VPN, your online activities are hidden even on public networks.

(ii) Disguising your whereabouts

VPN servers essentially act as your proxies on the internet. Because the demographic location data comes from a server in another country, your actual location cannot be determined. In addition, most VPN services do not store logs of your activities. Some providers, on the other hand, record your behavior, but do not pass this information on to third parties. This means that any potential record of your user behavior remains permanently hidden.

(iii) Access to regional content

Regional web content is not always accessible from everywhere. Services and websites often contain content that can only be accessed from certain parts of the world. Standard connections use local servers in the country to determine your location. This means that you cannot access content at home while traveling, and you cannot access international content from home. With VPN location spoofing , you can switch to a server to another country and effectively "change" your location.

(iv) Secure data transfer

If you work remotely, you may need to access important files on your company's network. For security reasons, this kind of information requires a secure connection. To gain access to the network, a VPN connection is often required. VPN services connect to private servers and use encryption methods to reduce the risk of data leakage.

Q10. Explain the evolutions of VPN's

Ans :

The history of VPNs

Since humans have been using the internet, there has been a movement to protect and encrypt internet browser data. The US Department of Defense already got involved in projects working on the encryption of internet communication data back in the 1960s.

The predecessors of the VPN

Their efforts led to the creation of ARPANET (Advanced Research Projects Agency Network), a packet switching network, which in turn led to the development of the Transfer Control Protocol/Internet Protocol (TCP/IP).

The TCP/IP had four levels: Link, internet, transport and application. At the internet level, local networks and devices could be connected to the universal network – and this is where the risk of exposure became clear. In 1993, a team from Columbia University and AT&T Bell Labs finally succeeded in creating a kind of first version of the modern VPN, known as SWIPE: Software IP encryption protocol.

In the following year, Wei Xu developed the IPSec network, an internet security protocol that authenticates and encrypts information packets shared online. In 1996, a Microsoft employee named Gurdeep Singh-Pall created a Peer-to-Peer Tunneling Protocol (PPTP).

Early VPNs

Contiguous to Singh-Pall developing PPTP, the internet was growing in popularity and the need for consumer-ready, sophisticated security systems emerged. At that time, anti-virus programs were already effective in preventing malware and spyware from infecting a computer system. However, people and companies also started demanding encryption software that could hide their browsing history on the internet.

The first VPNs therefore started in the early 2000s, but were almost exclusively used by companies. However, after a flood of security breaches, especially in the early 2010s, the consumer market for VPNs started to pick up.

VPNs and their current use

According to the Global Web Index, the number of VPN users worldwide increased more than fourfold between 2016 and 2018. In countries such as Thailand, Indonesia and China, where internet use is restricted and censored, one in five internet users uses a VPN. In the USA, Great Britain and Germany, the proportion of VPN users is lower at around 5%, but is growing.

One of the biggest drivers for VPN adoption in recent years has been the increasing demand for content with geographical access restrictions. For example, video streaming services such as Netflix or YouTube make certain videos available only in certain countries. With contemporary VPNs, you can encrypt your IP address so that you appear to be surfing from another country, enabling you to access this content from anywhere.

Q11. Explain different types of VPN's

Ans : **(Imp.)**

There are many different types of VPNs, but you should definitely be familiar with the three main types:

(i) SSL VPN

Often not all employees of a company have access to a company laptop they can use to work from home. During the corona crisis in Spring 2020, many companies faced the problem of not having enough equipment for their employees. In such cases, use of a private device (PC, laptop, tablet, mobile phone) is often resorted to. In this case, companies fall back on an SSL-VPN solution, which is usually implemented via a corresponding hardware box.

The prerequisite is usually an HTML-5-capable browser, which is used to call up the company's login page. HTML-5 capable browsers are available for virtually any operating system. Access is guarded with a user name and password.

(ii) Site-to-site VPN

A site-to-site VPN is essentially a private network designed to hide private intranets and allow users of these secure networks to access each other's resources.

A site-to-site VPN is useful if you have multiple locations in your company, each with its own local area network (LAN) connected to the WAN (Wide Area Network). Site-to-site VPNs are also useful if you have two separate intranets between which you want to send files without users from one intranet explicitly accessing the other.

Site-to-site VPNs are mainly used in large companies. They are complex to implement and do not offer the same flexibility as SSL VPNs. However, they are the most effective way to ensure communication within and between large departments.

(iii) Client-to-Server VPN

Connecting via a VPN client can be imagined as if you were connecting your home PC to the company with an extension cable. Employees can dial into the company network from their home office via the secure connection and act as if they were sitting in the office. However, a VPN client must first be installed and configured on the computer.

This involves the user not being connected to the internet via his own ISP, but establishing a direct connection through his/her VPN provider. This essentially shortens the tunnel phase of the VPN journey. Instead of using the VPN to create an encryption tunnel to disguise the existing internet connection, the VPN can automatically encrypt the data before it is made available to the user.

This is an increasingly common form of VPN, which is particularly useful for providers of insecure public WLAN. It prevents third parties from accessing and compromising the network connection and encrypts data all the way to the provider. It also prevents ISPs from accessing data that, for whatever reason, remains unencrypted and bypasses any restrictions on the user's internet access (for instance, if the government of that country restricts internet access).

The advantage of this type of VPN access is greater efficiency and universal access to company resources. Provided an appropriate telephone system is available, the employee can, for example, connect to the system with a headset and act as if he/she were at their company workplace. For example, customers of the company cannot even tell whether the employee is at work in the company or in their home office.

4.4 SMARTCARDS

Q12. Explain about Smart cards.

Ans :

A smartcard looks like a normal credit card with a chip embedded in it. Smartcards can be divided into three main categories according to the capabilities of the chip.

- Memory cards, which can just store data and have no data processing capabilities.
- Wired Logic Intelligent Memory cards, which contain also some built-in logic eliminating the need to insert and remove the card by hand. All that is necessary to start the interaction is to get close enough to a receiver. Contact less cards are practical in applications in which speed is important or in which card insertion/removal may be impractical (an example could be the Subscriber Identity Module (SIM) cards in mobile phones). Some manufacturers are making cards that function in both contact and contactless mode.

  All smartcards contain three types of memory: persistent non-mutable memory, persistent mutable memory and non-persistent mutable memory. ROM, EEPROM and RAM are the most widely used memories for the three respective types in the current smartcards.

  A typical processor card with contacts has 16KB ROM, 512 bytes of RAM and an eight-bit processor. Although the technology is moving towards 16 or 32-bit CPU (Oritz, 2003; ORACLE, 2010).

4.4.1 Application Security Using Smart cards

Q13. What are the Application Security Using Smart cards.

Ans : **(Imp.)**

A Smart Card is a plastic card the size of a credit card with an integrated circuit built into it. This integrated circuit may consist only of EEPROM in the case of a memory card, or it may also contain ROM, RAM and even a CPU.

Organizations are steadily migrating toward this technology. The days are numbered for a single mainframe used for computing every directive. Today, the delegation of tasks is being transferred to small, but dedicated smart cards. Their usefulness may soon exceed that of the standard computer for a variety of applications due, in part, to their portability and ease of use.

A smart card is a mini-computer without the display screen and keyboard. Smart cards contain a microchip with an integrated circuit capable of processing and storing thousands of bytes of electronic data. Due to the portability and size of smart cards they are seen as the next generation of data exchange.

Smart cards contain an operating system just like personal computers. Smart cards can store and process information and are fully interactive. Advanced smart cards also contain a file structure with secret keys and encryption algorithms. Due to the encrypted file system, data can be stored in separated files with full security.

Architecture

1. Most smart cards have been designed with the look and feel of a credit or debit card, but can function on at least three levels (credit - debit - personal information). Smart cards include a microchip as the central processing unit, random access memory (RAM) and data storage of around 10MB.
2. The smart card is an electronic recording device. Information in the microchip can instantaneously verify the card holder's identity and any privileges to which the card holder may be entitled. Information such as with drawals, sales, and bills can be processed immediately and if/when necessary; those records can be transmitted to a central computer for file updating.
3. Smart cards are secure, compact and intelligent data carriers. Smart cards should be regarded as specialized computers capable of processing, storing and safeguarding thousands of bytes of data.

4. Smart cards have electrical contacts and a thin metallic plate just above center line on one side of the card. Beneath this dime-sized plate is an integrated circuit (IC) chip containing a central processing unit (CPU), random access memory (RAM) and nonvolatile data storage.
5. Data stored in the smart card's microchip can be accessed only through the chip operating system (COS), providing a high level of data security. This security takes the form of passwords allowing a user to access parts of the IC chip's memory or encryption/decryption measures which translate the bytes stored in memory into useful information.
6. Smart cards typically hold 2,000 to 8,000 electronic bytes of data (the equivalent of several pages of data). Because those bytes can be electronically coded, the effective storage capacity of each card is significantly increased.
7. Magnetic-stripe cards, such as those issued by banks and credit card companies, lack the security of microchips but remain inexpensive due to their status as a single-purpose card.
8. Smart cards can be a carrier of multiple records for multiple purposes. Once those purposes are maximized, the smart card is often viewed as superior and, ultimately, less expensive.
9. The distributed processing possible with smart cards reduces the need for ever-larger mainframe computers and the expense of local and long-distance phone circuits required to maintain an on-line connection to a central computer.
10. Smart cards are defined by the ISO 7816 standards.

Security aspects

1. The microprocessor on the smart card is there for security. The host computer and card reader actually "talk" to the microprocessor.
2. The microprocessor enforces access to the data on the card. If the host computer read and wrote the smart card's random access memory (RAM), it would be no different than a diskette..
3. Smarts cards may have up to 8 kilobytes of RAM, 346 kilobytes of ROM 256 kilobytes of programmable ROM, and a 16-bit micro processor.
4. The smart card uses a serial interface and receives its power from external sources like a card reader. The processor uses a limited instruction set for applications such as cryptography.

Applications:

- Credit cards
- Electronic cash
- Computer security systems
- Wireless communication
- Loyalty systems (like frequent flyer points)
- Banking
- Satellite TV
- Government identification

Q14. Explain different types of Smart Cards.

Ans :

(i) Contact Cards and Contactless Cards

Contact Cards require insertion into a smart card reader with a direct connection to a conductive micro-module on the surface of the card.

Contact less Cards require only close proximity (a few inches) of a reader.

(ii) Dual-interface cards

Dual-interface cards are equipped with both contactless and contact interfaces. This type of card enables secure access to the smart card's chip with either the contactless or contact smart card interfaces.

(iii) Hybrid smart cards

Hybrid smart cards contain more than one smart card technology. For example, a hybrid smart card might have an embedded processor chip that is accessed through a contact reader and an RFID chip for proximity connection. The different chips may be used for different applications linked to a single smart card - for example, when a proximity chip is used for physical access control to restricted areas and a contact chip is used for SSO authentication.

(iv) Memory smart cards

Memory smart cards only contain memory chips and can only store, read and write data to the chip. The data on these cards can be overwritten or modified, but the card itself is not programmable. So, data can't be processed or modified programmatically. These cards can be read-only and used to store data such as a PIN, password or public key. They can also be read-write and used to write or update user data. Memory smart cards can be configured to be rechargeable or disposable, in which case the data they contain can only be used once or for a limited time before being updated or discarded.

(v) Microprocessor smart cards

Microprocessor smart cards have a microprocessor embedded onto the chip, in addition to memory blocks. A microprocessor card may also incorporate specific sections of files where each file is associated with a specific function. The data in the files and the memory allocation are managed with a smart card OS. This type of card can be used for more than one function and usually enables adding, deleting and otherwise manipulating data in memory.

Smart cards can also be categorized by their application, such as credit card, debit card, entitlement or other payment card, authentication token and so on.

4.4.2 Zero knowledge protocols and their use in smart cards

Q15. What is zero knowledge protocol.

(OR)

Explain the concept of zero knowledge protocol.

Ans : **(Imp.)**

Zero-knowledge is one of the most popular, useful and powerful protocol in cryptographic design which was introduced by Goldwasser ct al. (1985).

Zero-knowledge protocols, as their name suggest, arc cryptographic protocols in which one party (the prover) can demonstrate the knowledge of some secret to another party (the verifier) without revealing the secret. This way, an eavesdropper, as well as the verifier, can gain no information about the secret and cannot convince a third party that they know the secret. More precisely, the properties of a zero-knowledge protocol are as follows:

- The prover cannot cheat the verifier unless the prover is extremely lucky; By reiterat-ing the protocol, the odds of an impostor passing as a legitimate user can be made as minimal as necessary
- The verifier cannot pretend to be the prover to any third party because during the protocol execution the verifier gains no knowledge of the secret
- The verifier cannot convince a third party of the validity of the authentication proof.

A Basic Zero- Knowledge Protocol

The basic protocol consists of several rounds: what is explained below is repeated *n* times.

The prover uses the information he/she knows and a random number to transform the hard problem into another hard problem, one that is isomorphic to the original one. Not all problems and transformations, of course, are suitable for this purpose; the prover must be sure that the verifier cannot deduce any knowledge from the execution of the protocol, even after many iterations of it.

Then, the prover uses the information he/ she knows and the random number to solve the new instance of the hard problem, then commits to the solution, using a bit-commitment scheme. This kind of scheme is used when someone wants to commit to a result but does not want to reveal it until sometime later and. meanwhile; the counterpart wants to make sure that the result is not going to be changed after the commitment.

The prover reveals the new problem instance to the verifier, but the verifier cannot use this problem to get any information about the original instance or its solution. At this stage, the verifier asks the prover either to prove that the old and the new instances are isomorphic (i.e. two different solutions to two related problems) or to open the solution to which the prover committed before and show that it's a solution to the new instance. The prover complies.

Q16. State the advantages and disadvan- tages of smart cards.

Ans : **(Imp.)**

Advantages

Smart cards offer several advantages, such as these:

- **Stronger security**

  Smart cards provide a higher level of security than magnetic stripe cards because they contain microprocessors capable of processing data directly without remote connections. Even memory-only smart cards can be more secure because they can store more authentication and account data than traditional mag stripe cards. Smart cards are generally safe against electronic inter- ference and magnetic fields, unlike magnetic stripe cards.

- **Information persistence**

  Once information is stored on a smart card, it can't be easily deleted, erased or altered. That is why smart cards are good for storing valuable data that should not be reproduced. However, applications and data on a card can be updated through secure channels, so issuers do not have to issue new cards when an update is needed.

Disadvantages

While smart cards have many advantages, there also drawbacks, including the following:

- **Cost**

  The cards and the smart card readers can be expensive.

- **Compatibility**

  Not all smart card readers are compatible with all types of smart cards. Some readers use nonstandard protocols for data storage and card interface, and some smart cards and readers use proprietary software that is incompatible with other readers.

- **Security vulnerabilities**

  Smart cards are secure for many applications, but they are still vulnerable to certain types of attack. For example, attacks that can recover information from the chip can target smart card technology. Differential power analysis (DPA) can be used to deduce the on-chip private key used by public key algorithms, such as the Rivest-Shamir-Adleman (RSA) algorithm. Some implementations of symmetric ciphers are vulnerable to timing attacks or DPA. Smart cards may also be physically disassembled in order to gain access to the onboard microchip.

Examples

Examples of smart card applications include the following:

- Payment cards, including debit and credit cards issued by commercial credit card companies and banks, are used for financial transactions.
- Electronic benefits transfer cards are used for distribution of government benefits, such as the U.S. Supplemental Nutrition Assistance Program.
- Transit cards let local and regional transit systems process payments, as well as give riders points on their purchases.
- Access control cards enable schools, companies and government entities to control access to physical locations.
- Smart health cards help medical facilities securely store patient medical records.
- SIM cards, used inside of digital cameras and smart phones, store media and other data.

4.4.3 Attacks on Smart Cards

Q17. Explain different types of attacks on Smart cards.

Ans : **(Imp.)**

There are three main types of attack that are considered in smart card security. These are:

1. Invasive Attacks

These are attacks that require the micro-processor in a smart card to be removed and directly attacked through a physical means. This class of attacks can, at least in theory, compromise the security of any secure microprocessor. However, these attacks typically require wry exp ensive equipment and a large investment in time to produce results. Invasive attacks are therefore considered to be primarily in the realm of semiconductor manufacturers and students at well-funded universities.

An example of such an attack would be to place probes on bus lines between blocks of a chip (a hole needs to be made in the chip's passivation layer to allow this). An attacker could then attempt to derive secret information by observing the information that is sent form one block to another.

At its most extreme this type of attack could make use of a focused ion beam to destroy or create tracks on the chips surface. In fuse was present. Once the fuse was blown inside the chip (before the chip left the manufacturer's factory) this mode was no longer available. In modern secure microprocessors this test circuit is typically removed when the chip is cut from the die and this attack is no longer possible.

2. Semi-Invasive Attacks

These attacks require the surface of the chip to be exposed. An attacker then seeks to compromise the security of the secure micro- processor without directly modifying the chip.

3. Non-Invasive Attacks

These attacks seek to derive information without modifying a smart card, i.e. both the secure micro-processor and the plastic card remain unaffected. An attacker will attempt to derive information by observing information that leaks during the computation of a given command, or attempt to inject faults using mechanisms other than light.

UNIT V

Applications: Kerberos, Web Security Protocols (SSL), IPSec, Electronic Payments, E-cash, Secure Electronic Transaction (SET), Micro Payments, Case Studies of Enterprise Security (.NET and J2EE)

5.1 KERBEROS

Q1. Write a short note on kerberos.

(OR)

Explain about kerberos.

Ans :

Kerberos is a network authentication protocol. It is designed to provide strong authentication for client/server applications by using secret-key cryptography. A free implementation of this protocol is available from the Massachusetts Institute of Technology Kerberos is available in many commercial products as well.

The Internet is an insecure place. Many of the protocols used in the Internet do not provide any security. Tools to "sniff" passwords off of the network are in common use by malicious hackers. Thus, applications which send an unencrypted password over the network are extremely vulnerable. Other applications rely on the client to restrict its activities to those which it is allowed to do, with no other enforcement by the server. Some sites attempt to use firewalls to solve their network security problems. Unfortunately, firewalls assume that the hackers are always from the outside, which is often a very bad assumption.

Most of the really damaging incidents of computer crime are carried out by insiders. Firewalls also have a significant disadvantage in that they restrict the usage of internet by the users.

Kerberos was created by MIT as a solution to these network security problems. The Kerberos protocol uses strong cryptography so that a client can prove its identity to a server (and vice versa) across an insecure network connection. After a client and server have used Kerberos to prove their identity, they can also encrypt all of their communications to assure privacy and data integrity as they go about their business.

The first published report on Kerberos listed the following requirements;

1. **Secure:** Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
2. **Reliable:** Kerberos should be highly reliable and should employ distributed server architecture, with one system able to back up another.
3. **Transparent:** The user should not be aware that authentication is taking place, beyond the requirement to enter a password.
4. **Scalable:** The system should be capable of supporting large numbers of clients and servers.

Q2. Explain about the various keys used with kerberos.

Ans : **(Imp.)**

1. **Keys Privacy through Encryption:** Kerberos messages are encrypted with various encryption keys to ensure that no one can tamper with the client's ticket or with other data in a Kerberos message. Possible Kerberos keys include:
2. **Long-term key:** This is a key which is known only to the target server and the KDC - with which the client's ticket is encrypted.
3. **Client/server session key:** A short-term, single-session key that is created by the KDC and used to encrypt the client-to-server and server-to-client messages after identity and authorization have been confirmed.

4. KDC/user session key: The KDC and the user share a secret encryption key as well, which is used, for example, to encrypt the message to the client containing a session key.

Q3. Explain the mechanism of kerberos.

Ans :

The Kerberos Authentication System uses a series of encrypted messages to prove to a verifier that a client is running on behalf of a particular user. The Kerberos protocol is based in part on the Needham and Schroeder authentication protocol but with changes to support the needs of the environment for which it was developed. Among these changes are the use of timestamps to reduce the number of messages needed for basic authentication, the addition of a "ticket-granting" service to support subsequent authentication without reentry of a principal's password, and different approach to cross-realm authentication (authentication of a principal registered with a different authentication server than the verifier).

The remainder of this section describes the Kerberos protocol. The description is simplified for clarity; additional fields are present in the actual protocol. Readers should consult RFC 1510 for a more thorough description of the Kerberos protocol.

Q4. Explain the mechanism of kerberos ticket.

Ans : **(Imp.)**

The client and server do not initially share an encryption key. Whenever a client authenticates itself to a new verifier it relies on the authentication server to generate a new encryption key and distribute it securely to both parties. This new encryption key is called a session key and the Kerberos ticket is used to distribute it to the verifier.

The Kerberos ticket is a certificate issued by an authentication server encrypted using the server key. Among other information, the ticket contains the random session key that will be used for authentication of the principal to the verifier, the name of the principal to whom the session key was issued, and an expiration time after which the session key is no longer valid. The ticket is not sent directly to the verifier, but is instead sent to the client who forwards it to the verifier as part of the application request. Because the ticket is encrypted in the server key, known only by the authentication server and intended verifier, it is not possible for the client to modify the ticket without detection.

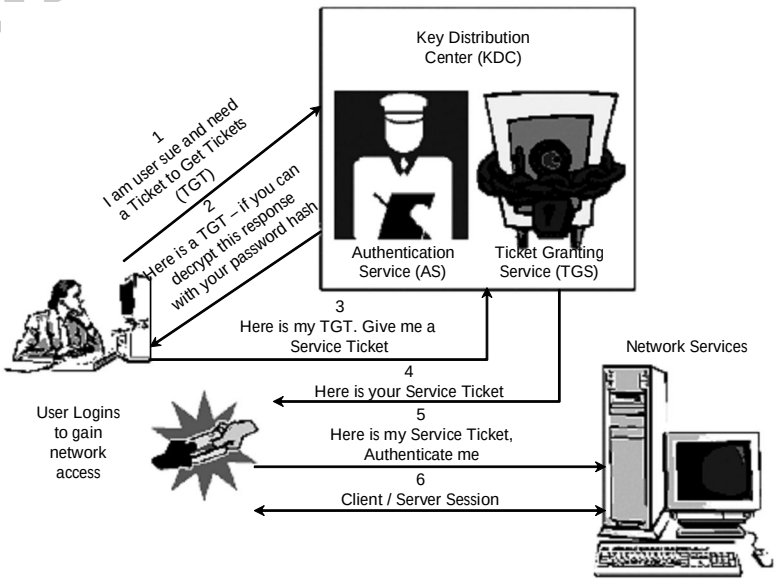
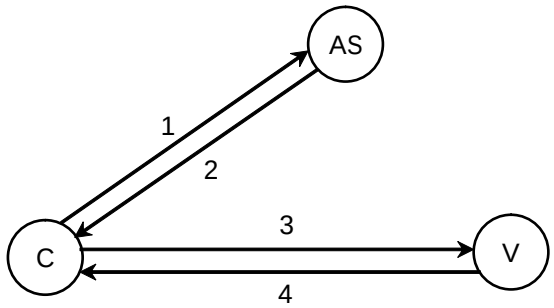


Fig.: Kerberos Ticket Exchange

(i) Application request and response

Messages 3 and 4 in figure 1 show the application request and response, the most basic exchange in the Kerberos protocol. It is through this exchange that a client proves to a verifier that it knows the session key embedded in a Kerberos ticket. There are two parts to the application request, a ticket (described above) and an authenticator. The authenticator includes, among other fields: the current time, a checksum, and an optional encryption key, all encrypted with the session key from the accompanying ticket.



1. As_req: c, v, $time_{exp}$, n
2. As_rep: {$K_{o,v}$, v, $time_{exp}$, n, ...} K_o, {$T_{o,v}$}Kv
3. ap_req: {ts, ck, Ksubsession, ...} $K_{e,v}$ {$T_{o,v}$}Kv
4. ap_rep: {ts} $K_{o,v}$ (Optional)

   $T_{o,v}$ = $K_{o,v,\,c,}$ $time_{exp}$ ...

Fig.: Basic Kerberos authentication protocol (simplified)

Upon receipt of the application request, the verifier decrypts the ticket, extracts the session key, and uses the session key to decrypt the authenticator. If the same key was used to encrypt the authenticator as used to decrypt it, the checksum will match and the verifier can assume the authenticator was generated by the principal named in the ticket and to whom the session key was issued. This is not by itself sufficient for authentication since an attacker can intercept an authenticator and replay it later to impersonate the user. For this reason the verifier additionally checks the timestamp to make sure that the authenticator is fresh. If the timestamp is within a specified window (typically 5 minutes) centered around the current time on the verifier, and if the timestamp has not been seen on other requests within that window, the verifier accepts the request as authentic.

At this point the identity of the client has been verified by the server. For some applications the client also wants to be sure of the server's identity. If such mutual authentication is required, the server generates an application response by extracting the client's time from the authenticator, and returns it to the client together with other information, all encrypted using the session key.

(ii) Authentication request and response

The client requires a separate ticket and session key for each verifier with which it communicates. When a client wishes to create an association with a particular verifier, the client uses the authentication request and response, messages 1 and 2 from figure 1, to obtain a ticket and session key from the authentication server. In the request, the client sends the authentication server its claimed identity, the name of the verifier, a requested expiration time for the ticket, and a random number that will be used to match the authentication response with the request.

In its response, the authentication server returns the session key, the assigned expiration time, the random number from the request, the name of the verifier, and other information from the ticket, all encrypted with the user's password registered with the authentication server, together with a ticket containing similar information, and which is to be forwarded to the verifier as part of the application request. Together, the authentication request and response and the application request and response comprise the basic Kerberos authentication protocol.

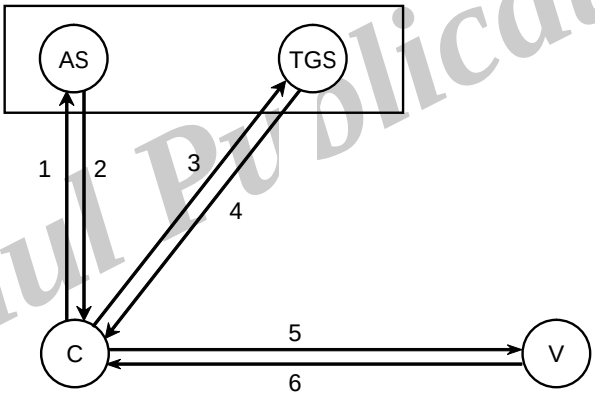
(iii) Obtaining additional tickets

The basic Kerberos authentication protocol allows a client with knowledge of the user's password to obtain a ticket and session key for and to prove its identity to any verifier registered with the authentication server. The user's password must be presented each time the user performs authentication with a new verifier. This can be cumbersome; instead, a system should support single sign-on, where the user logs in to the system once, providing the password at that time, and with subsequent authentication occurring automatically. The obvious way to support this, caching the user's password on the workstation, is dangerous. Though a Kerberos ticket and the key associated with it are

valid for only a short time, the user's password can be used to obtain tickets, and to impersonate the user until the password is changed. A better approach, and that used by Kerberos, is to cache only tickets and encryption keys (collectively called credentials) that will work for a limited period.

The ticket granting exchange of the Kerberos protocol allows a user to obtain tickets and encryption keys using such short-lived credentials, without re-entry of the user's password. When the user first logs in, an authentication request is issued and a ticket and session key for the ticket granting service is returned by the authentication server. This ticket, called a *ticket granting ticket*, has a relatively short life (typically on the order of 8 hours). The response is decrypted, the ticket and session key saved, and the user's password forgotten.

Subsequently, when the user wishes to prove its identity to a new verifier, a new ticket is requested from the authentication server using the ticket granting exchange. The ticket granting exchange is identical to the authentication exchange except that the ticket granting request has embedded within it an application request, authenticating the client to the authentication server, and the ticket granting response is encrypted using the session key from the ticket granting ticket, rather than the user's password. Figure 2 shows the complete Kerberos authentication protocol. Messages 1 and 2 are used only when the user first logs in to the system, messages 3 and 4 whenever a user authenticates to a new verifier, and message 5 is used each time the user authenticates itself. Message 6 is optional and used only when the user requires mutual-authentication by the verifier.



1. as_req: c, tgs, $time_{exp}$, n
2. as_rep: $\{K_{c,\ tgs}$, tgs, $time_{exp}$, n, ...$\}K_c$, $\{T_{c,tgs}\}K_{tgs}$
3. tgs_req: {ts, ...}$K_{c,tgs}$ $\{T_{c,tgs}\}$ $\{T_{c,tgs}\}K_{tgs}$, v, $time_{exp}$, n
4. tgs_rep: $\{K_{c,v,}$ V, $time_{exp}$, n, ...$\}$ $K_{c,tgs,}$ $\{T_c$, v$\}$ K_v
5. ap_req: {ts, ck, K subsession, ...}$K_{c,v}$ $\{T_{c,v}\}$Kv
6. ap_rep: {ts}K c,v (optional)

Fig. 2: Complete Kerberos Authentication Protocol (simplified)

Q5. What are the various types of tickets with kerberos.

Ans :

This is physically securing the node with complete authentication database. The KDC shares a secret key, known as a master key, with each principal. When Alice informs the KDC that she wants to talk to Bob, the KDC invents a session key Kab for Alice and Bob to share, encrypts Kab with Alice's master key

for Alice, encrypts Kab with Bob's master key for Bob, and returns all this information to Alice. The message consisting of the session key Kab encrypted with Bob's master key is known as a ticket to Bob. The session key Kab together with the ticket to Bob are known as Alice's credentials to Bob.

1. Ticket Granting Ticket(TGT)

During the initial login, the workstation, on Alice's behalf, asks the KDC for a session key for Alice. The KDC generates a session key Sa, and transmits Sa to the workstation. The KDC also sends a ticket granting ticket which is Sa encrypted with the KDC's master key.

2. Ticket Granting Server (TGS)

Other than TGT's which are received from KDC, the other tickets are all received from the TGS. TGS has to have the same database as the KDC. So in Kerberos, the TGS and KDC is really same thing. It would be easier to understand Kerberos if the documentation didn't refer to two different entities and two sets of protocol messages that basically do the same thing.

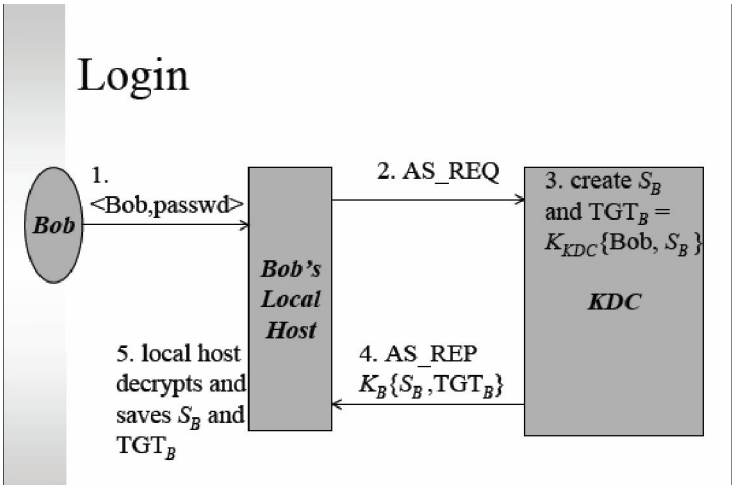
3. Session Key and Ticket-granting Ticket (TGT)

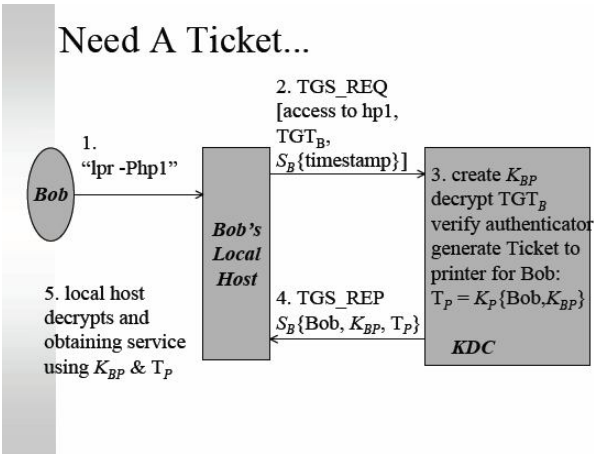
The messages between a host and the KDC can be protected using the principal's master key. For every request to KDC from the principal needs the session key. It insists on principal retyping in the password. It is important to remember the principal's password and the principal's master key derived from the password.

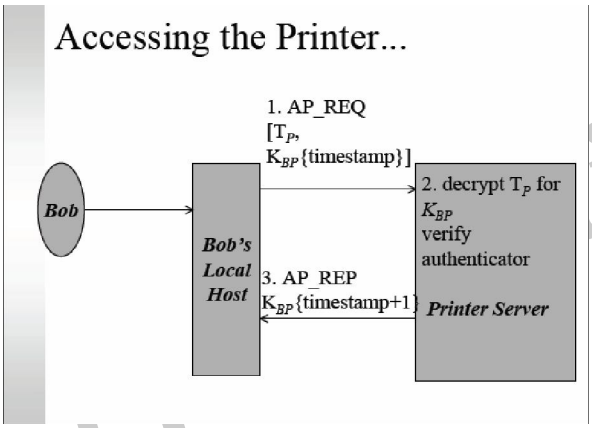
To avoid potentially too much exposure to password/master key it is necessary at initial login, a per principal session key *SB* (for Bob) is requested from KDC.*SB* has a limited valid time period. A TGT for Bob is also issued by the KDC, which includes the session key *SB* and Bob's identification information; all encrypted using the KDC's master key.

Bob's Kerberos client (e.g., the login host) decrypts and remembers the SB, for subsequent message with KDC and the TGT, for reminding/convincing KDC to use SB with it as well. It is not necessary to remember or store the password. Every new request to KDC must include TGT in the request message and the new tickets from KDC must be decrypted with SB.

Login







Realms

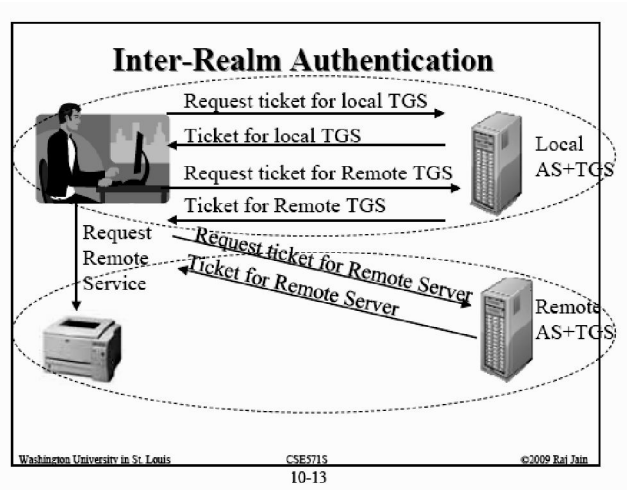
It is very difficult to have a single KDC for an entire network. With replicated KDC's the bottleneck problems can be eliminated and the single point of failure can be easily identified. But whoever manages the KDC can access every user's master key, and therefore access everything that every user can access. Everyone must also trust the physical controls around every replica of the KDC, and there would have to be widely dispersed replicas to ensure availability and convenience.

Because of this reason the network are divided into realms. Each and every realm has its own KDC database. There can be multiple KDC's in a realm, but they would be equivalent, have the same KDC, aster key, and have the same database of principal's master keys. Two KDC/s in different realms, would have different KDC master keys and totally different principals master key databases, since they would be responsible for a different set of principals.

Key components

1. Realm = One organization or one trust domain
2. Each realm has its own set of principles including KDC/TGT
3. Each Principal's name = Name + Instance + Realm
4. Each principle is of 40 characters each with null terminated.
5. Instance = Particular Server or Human role (administrator, game player)
6. In V4, both realms should have a direct trust relationship and chaining prohibited

Interrealm authentication



Q6. What are the various types of tickets with kerberos V5?

Ans : **(Imp.)**

I) Name

1. Kerberos V5 contains two components: REALM and the NAME.
2. The Name component contains a type and a varying number of arbitrary strings.
3. This arbitrary string field serves the purpose of INSTANCE field with kerberos4.
4. The REALMS field can be DNS standard names or X.500 names, and the syntax allows for other name types as well.

II) Delegation of rights

One way delegation could be provided is for Alice to send bob her master key (in an encrypted message to bob.), allowing him to obtain tickets to whatever resources he might need on Alice behalf. That is clearly not desirable, since it would allow bob forever be able to impersonate Alice. These mechanisms are inconvenient and /or insecure and therefore undesirable but they wouldn't work with Kerberos (either V4 or V5). Kerberos V5 explicitly allows delegation by allowing Alice to ask for TGT with a network layer address different from hers. As a matter of fact it allows Alice to specify multiple addresses to include (in which case the ticket can be used form any of the specified addresses), or allows Alice to request that no address be included (in which case the ticket can be used for any address.) Kerberos V5 supports two forms of limited delegations:

1. Alice can give Bob tickets to the specific services he will need to access on her behalf (rather than giving him a TGT, which will allow him to request tickets to any services.)
2. When requesting a ticket or TGT that she intends to give Bob, Alice can request that the field AUTHORIZATION-DATA be added to the ticket or TGT.

The fields is not interpreted by Kerberos, but it instead application-specific, which means it is left up to the application to define and use the field. The intention is that the field specifies to the application restrictions on what Bob is allowed to do with the ticket. If the field is a TGT Alice gives to bob, the field will be copied by the KDC into any ticket bob gets using that TGT. OSF/DSE security and windows 2000 make extensive use of this field. There are two flags in a TGT involving delegation permission.

One indicates that a TGT is **forward able**, which means that it can be exchanged for a TGT with a different layer address. This gives Alice permission to give bob a TGT, with which bob can request ticket to any resources on Alice behalf. When Alice uses a forward able TGT to request a TGT to be used from bobs network layer address, she also specifies how the FORWARDABLE flag set, and then bob will be able to use that TGT to obtain a TGT from some other entity Carol, allowing Carol to act on Alice's behalf. The other flag that the TGT is proxiable, meaning that it can be used to request tickets for use what a different network layer address than the one in the TGT. This gives Alice permission to get tickets that she can give to Bob, but not a TGT for use by bob. The Kerberos documentation refers to tickets Alice gives to bob for use on her behalf for her **PROXY TICKETS**.

III) Ticket Lifetime

In Kerberos V5, tickets can be issued with virtually unlimited life times. The time stamp format is an ASN.1- defined quantity that is 17 octets long. Although it has a virtually unlimited lifetime, it is only in seconds, and Kerberos V5, in some cases, would have preferred time expressed down to micro-seconds. Long-lived tickets pose serious security risk, because once created they cannot be revoked. So V5 has a number of mechanisms for implementing revocable long-lived tickets. These mechanisms involve use of several timestamp fields in tickets.

1. START-TIME – time the ticket becomes valid.
2. END-TIME – time the ticket expires.
3. AUTHTIME – time at which Alice first logged in, that is, when she was granted an initial TGT. AUTHTIME is copied from a initial TGT into each ticket she requests based on the TGT.
4. RENEW-TILL – latest legal end-time.

(i) Renewable Tickets

Rather than creating a ticket valid for say, 100 years, the KDC can give Alice a ticket that will be valid for 100years, but only if she keeps renewing it, say once a day. Renewing a ticket involves giving the ticket to the KDC and having the KDC reissue it. If there is some reason to want to revoke Alice privileges, this can be done by telling the KDC not to renew any of the Alice's tickets. The KDC is configured with the maximum validity time for a ticket, say a day. If there is a reason for Alice's ticket to be valid for longer than that time, then when Alice requests the ticket, the KDC sets the RENEWABLE flag inside the ticket. The RENEW-TILL time specifies the time beyond which the ticket cannot be renewed. The END-TIME specifies the time at which the ticket will expire. When ALICE gives the KDC a renewable ticket and requests that it be renewed, the KDC does this by changing END-TIME to be the maximum ticket lifetime as configured into the users entry in the KDC database, added to the current time.

(ii) Postdated Tickets

Postdated tickets are used to run a batch job at sometime in the future. Suppose you want to issue a ticket starting a week from now and good for two hours. One possible method is to issue a ticket with an expiration time of one week plus two hours from the present time, but that would mean the ticket would be valid form the time it was issued until it expired. Kerberos instead allows a ticket to become valid at some point in the future. Kerberos does this by using the START-TIME timestamp, indicating when the ticket should first become valid. Such a ticket is known as postdated ticket. In order to allow a revocation of the postdated ticket between the time it was issued and the time it became valid, there's an invalid flag inside the ticket that the Kerberos sets in the initially issued postdated ticket. When the time specified START-TIME occurs, Alice can present the ticket to the KDC and the KDC will clear the invalid flag. This additional step gives the opportunity to revoke the postdated ticket by warning the KDC. If the KDC is configured to revoke the postdated ticket, the validation request will fail.

Q7. State the advantages of kerberos?

Ans :

Advantages of Kerberos

- User's passwords are never sent across the network, encrypted or in plain text Secret keys are only passed across the network in encrypted form
- Client and server systems mutually authenticate limits the duration of their users' authentication.
- Authentications are reusable and durable
- Kerberos has been scrutinized by many of the top programmers, cryptologists and security experts in the industry.

5.2 Web Security Protocols

Q8. Explain briefly about Web Security Protocols.

Ans :

Usage of internet for transferring or retrieving the data has got many benefits like speed, reliability, security etc. Much of the Internet's success and popularity lies in the fact that it is an open global network. At the same time, the fact that it is open and global makes it not very secure. The unique nature of the Internet makes exchanging information and transacting business over it inherently dangerous. The faceless, voiceless, unknown entities and individuals that share the Internet may or may not be who or what they profess to be. In addition, because the Internet is a global network, it does not recognize national borders and legal jurisdictions. As a result, the transacting parties may not be where they say they are and may not be subject to the same laws or regulations.

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

5.2.1 SSL

Q9. Explain the concept of SSL

Ans : **(Imp.)**

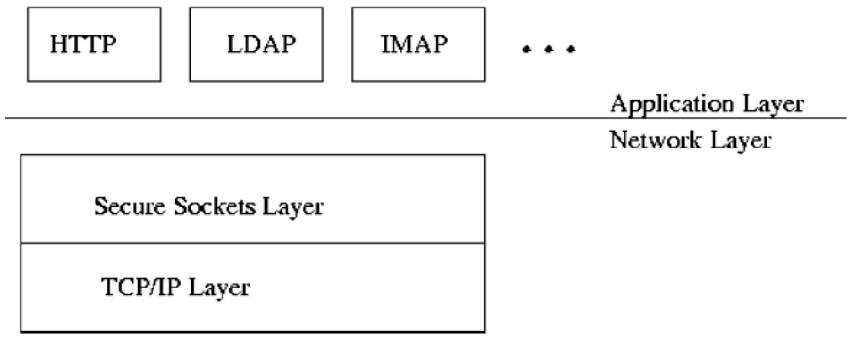
IPSec provides security at the network level and the main advantage is that it is transparent to end users and applications. In addition, IPSec includes a filtering capability so that only selected traffic can be processed. Secure Socket Layer or Transport Layer Security (SSL/TLS) provides security just above the TCP at transport layer. Two implementation choices are present here. Firstly, the SSL/TLS can be implemented as a part of TCP/IP protocol suite, thereby being transparent to applications.

Alternatively, SSL can be embedded in specific packages like SSL being implemented by Netscape and Microsoft Explorer browsers. Secure Electronic Transaction (SET) approach provides application-specific services i.e., according to the security requirements of a particular application. The main advantage of this approach is that service can be tailored to the specific needs of a given application.

Secure Socket Layer / Transport Layer Security

SSL was developed by Netscape to provide security when transmitting information on the Internet. The Secure Sockets Layer protocol is a protocol layer which may be placed between a reliable connection-oriented network layer protocol (e.g. TCP/IP) and the application protocol layer (e.g. HTTP).

SSL runs above TCP/IP and below high-level application protocols



SSL provides for secure communication between client and server by allowing mutual authentication, the use of digital signatures for integrity and encryption for privacy. SSL protocol has different versions such as SSLv2.0, SSLv3.0, where SSLv3.0 has an advantage with the addition of support for certificate chain loading. SSL 3.0 is the basis for the Transport Layer Security [TLS] protocol standard. 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 shown below:

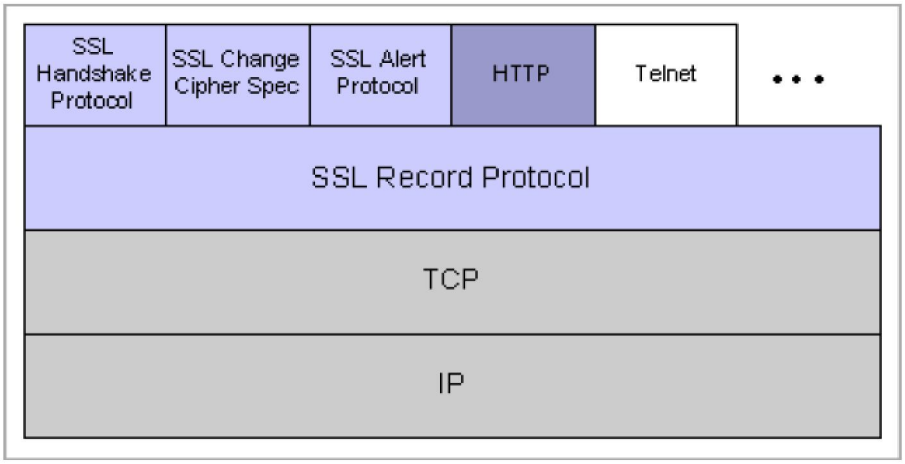


Fig.: SSL Protocol Stack

The SSL Record Protocol provides basic security services to various higher-layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/ server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: the Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol. 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. An SSL session is *stateful*. Once a session is established, there is a current operating state for both read and write (i.e., receive and send). In addition, during the Handshake Protocol, pending read and write states are created. Upon successful conclusion of the Handshake Protocol, the pending states become the current states.

An SSL session may include multiple secure connections; in addition, parties may have multiple simultaneous sessions.

A session state is defined by the following parameters:

- **Session identifier**: An arbitrary byte sequence chosen by the server to identify an active or resumable session state.
- **Peer certificate: An** X509.v3 certificate of the peer. This element of the state may be null.
- **Compression method**: The algorithm used to compress data prior to encryption.
- **Cipher spec: Specifies** the bulk data encryption algorithm (such as null, AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash_size.

- **Master secret: 48-byte** secret shared between the client and server.
- **Is resumable: A** flag indicating whether the session can be used to initiate new connections.

A connection state is defined by the following parameters:

- **Server and client random**: Byte sequences that are chosen by the server and client for each connection.
- **Server write MAC secret**: The secret key used in MAC operations on data sent by the server.
- **Client write MAC secret**: The secret key used in MAC operations on data sent by the client.
- **Server write key: The** conventional encryption key for data encrypted by the server and decrypted by the client.
- **Client write key**: The conventional encryption key for data encrypted by the client and decrypted by the server.
- **Initialization vectors:** When a block cipher in CBC mode is used, an initialization vector (IV) is maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter the final ciphertext block from each record is preserved for use as the IV with the following record.
- **Sequence numbers**: Each party maintains separate sequence numbers for transmitted and received messages for each connection. When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed 264-1.

5.3 IPSEC

Q10. Explain the concept of IPSec.

Ans : **(Imp.)**

Definition

Internet Protocol security (IPSec) is a framework of open standards for protecting communications over Internet Protocol (IP) networks through the use of cryptographic security services. IPSec supports network-level peer authentication, data origin authentication, data integrity, data confidentiality (encryption), and replay protection.

Need

In Computer Emergency Response Team (CERT)'s 2001 annual report it listed 52,000 security incidents in which most serious types of attacks included IP spoofing, in which intruders create packets with false IP addresses and exploit applications that use authentication based on IP and various forms of eavesdropping and packet sniffing, in which attackers read transmitted information, including logon information and database contents. In response to these issues, the IAB included authentication and encryption as necessary security features in the next-generation IP i.e. IPv6.

Applications of IPSec

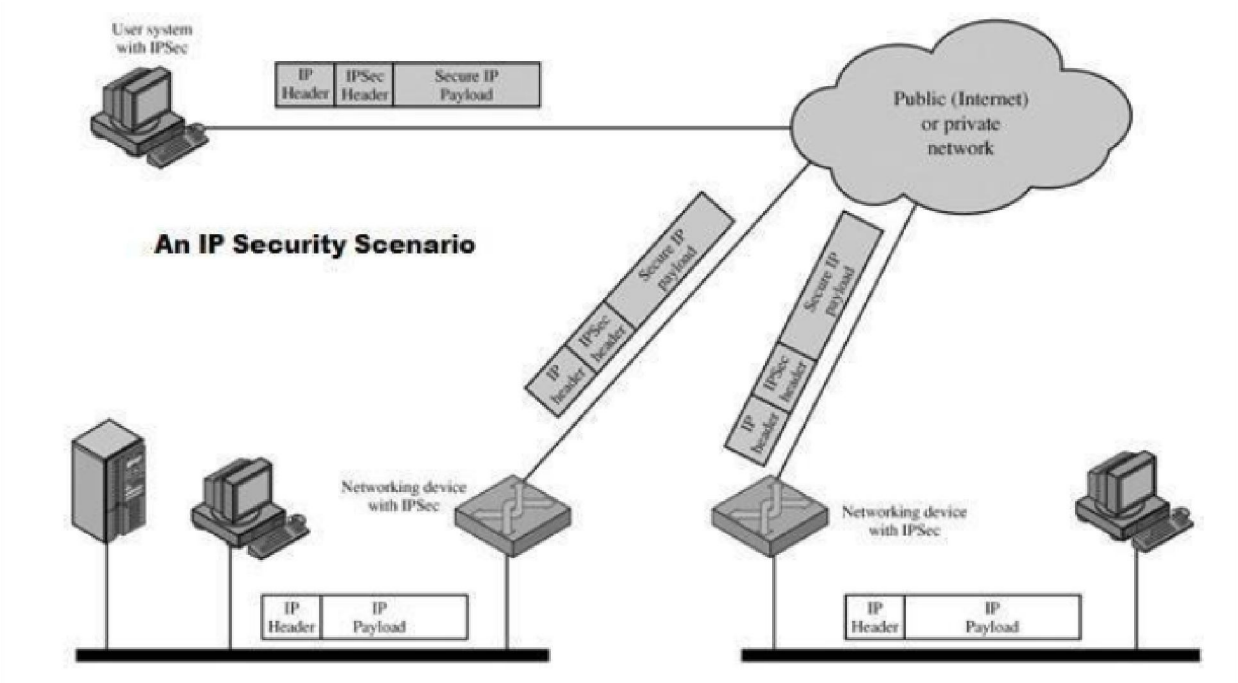
IPSec provides the capability to secure communications across a LAN, across private and public wide area networks (WAN's), and across the Internet.

- **Secure branch office connectivity over the Internet**: A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead.

- **Secure remote access over the Internet**: An end user whose system is equipped with IP security protocols can make a local call to an Internet service provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for travelling employees and telecommuters.
- **Establishing extranet and intranet connectivity with partners**: IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.
- **Enhancing electronic commerce security**: Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security.

The principal feature of IPSec enabling it to support varied applications is that it can encrypt and/or authenticate all traffic at IP level. Thus, all distributed applications, including remote logon, client/server, e-mail, file transfer, Web access, and so on, can be secured.

The following figure shows a typical scenario of IPSec usage. An organization maintains LANs at dispersed locations. Non secure IP traffic is conducted on each LAN.



The IPSec protocols operate in networking devices, such as a router or firewall that connect each LAN to the outside world. The IPSec networking device will typically encrypt and compress all traffic going into the WAN, and decrypt and decompress traffic coming from the WAN; these operations are transparent to workstations and servers on the LAN. Secure transmission is also possible with individual users who dial into the WAN. Such user workstations must implement the IPSec protocols to provide security.

Benefits

The benefits of IPSec are listed below:

- IPSec in a firewall/router provides strong security to all traffic crossing the perimeter
- IPSec in a firewall is resistant to bypass
- IPSec is below transport layer (TCP, UDP), hence transparent to applications
- IPSec can be transparent to end users
- IPSec can provide security for individual users if needed (useful for offsite workers and setting up a secure virtual subnetwork for sensitive applications).

5.4 ELECTRONIC PAYMENTS

Q11. Explain the concept of Electronic Payments.

Ans :

Meaning

An e-payment system is a way of making transactions or paying for goods and services through an electronic medium, without the use of checks or cash.

It's also called an electronic payment system or online payment system. Read on to learn more.

The electronic payment system has grown increasingly over the last decades due to the growing spread of internet-based banking and shopping.

As the world advances more with technology development, we can see the rise of electronic payment systems and payment processing devices.

Electronic Payment Methods

One of the most popular payment forms online is credit and debit cards.

Besides them, there are also alternative payment methods, such as bank transfers, electronic wallets, smart cards, or bitcoin wallets (bitcoin is the most popular cryptocurrency).

E-payment methods could be classified into credit payment systems and cash payment systems.

1. Credit Payment System

Credit Card: A form of the e-payment system which requires the use of the card issued by a financial institute to the cardholder. The cards are for making payments online or through an electronic device, without the use of cash.

Still, one of the most popular e-payment methods are credit and debit card #payments

Click to Tweet

E-wallet: A form of prepaid account that stores user's financial data, like debit and credit card information. Invented to make an online transaction easier.

Smart card: A plastic card with a microprocessor that can be loaded with funds to make transactions. Also known as a chip card.

2. Cash Payment System

Direct debit: A financial transaction in which the account holder instructs the bank to electronically collect a specific amount of money from his account to pay for goods or services.

E-check: A digital version of an old paper check. It's an electronic transfer of money from a bank account, usually checking account, without the use of the paper check.

E-cash: A form of an electronic payment system, where a certain amount of money is stored on a client's device and made accessible for online transactions.

Stored-value card: A card with a certain amount of money that can be used to perform the transaction in the issuer store. A typical example of stored-value cards are gift cards.

Pros of Using an E-payment System

E-payment systems are made to facilitate electronic payments for online transactions. With the growing popularity of online shopping, e-payment systems have become a must for online consumers — to make shopping and banking more convenient.

It comes with many benefits, such as:

- Reaching more clients from all over the world, which results in more sales.
- More effective and efficient transactions – It's because transactions are made in seconds (with one-click), without wasting customer's time. It comes with speed and simplicity.
- Convenience. Customers can pay for items on an e-commerce website at anytime and anywhere. They just need an internet connected device. As simple as that!
- Expenses control for customers, as they can always check their virtual account where they can find the transaction history.
- Today it's easy to add payments to a website, so even a non-technical person may implement it in minutes and start processing online payments.
- Payment gateways and payment providers offer highly effective security and anti-fraud tools to make transactions reliable.

5.5 E-CASH

Q12. What is E-cash? Explain the implementation of E-cash?

Ans : **(Imp.)**

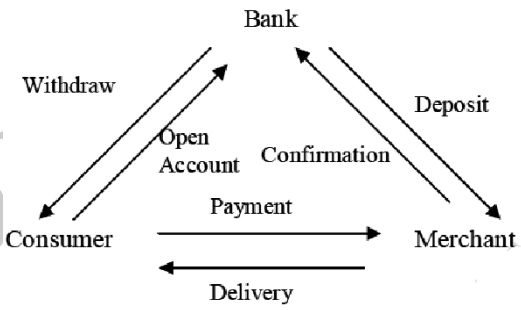
Meaning

E-cash is one of the services that attract people attention for doing business transaction electronically. It is a replacement for traditional coins and paper note.

Implementation

1. Consumer needs to open an account with a bank. Merchant who wants to participate in E-cash transaction need to have accounts with various banks in order to support consumer's transaction who might use any bank account. The banks on the other hand will handle both consumers' and merchants' accounts.
2. When consumer decides to purchase, he or she will transfers the E-cash from his/her bank account to his/her electronic purse (on-line system) or E-cash token (off-line system). The E-cash can then be transferred to the merchant in exchange with the merchant's products or services. The E-cash payment can be in term of softcopy (via software) or token based. Transactions via Internet are normally encrypted.
3. Upon receiving E-cash payment from consumer, merchant will get confirmation from the bank. The bank will then authenticate the E-cash transaction. At the same time the bank will debit consumer's account based on the agreed amount. The merchant will then delivers the products or services and instructs the bank to deposit the agreed amount to the merchant's bank account.

The diagram below represent E-cash processes in general:



Q13. Explain the properties of E-cash.

Ans :

To be able to replace coins and paper notes, E-cash should be as good as coins and paper notes in term of features. Some of the important features of coins and paper notes are: transferable, acceptable, dividable, untraceable and anonymous. Listed below are some of the important properties for E-cash implementation. Later discussions on E-cash implementations will be based on these few properties.

1. Security

For any E-cash system to be accepted, security is one of the prime concerns that need to be considered. The originality of the message being transferred among consumers, merchants and banks need to be secured to avoid any unauthorized individual intercepting or changing the content of the messages. In order to protect E-cash from

such illegal activity, E-cash system must possess quality such as integrity, nonrepudiation and able to authenticate. All parties must know to whom they are dealing with, before engaging or committing in any transaction. Integrity comes in place where the message sent by consumers, merchants and banks must be intact when it reaches respective recipients. Once the integrity and authentication are achieved, consumers, merchants or banks could no longer deny the transaction.

2. Privacy

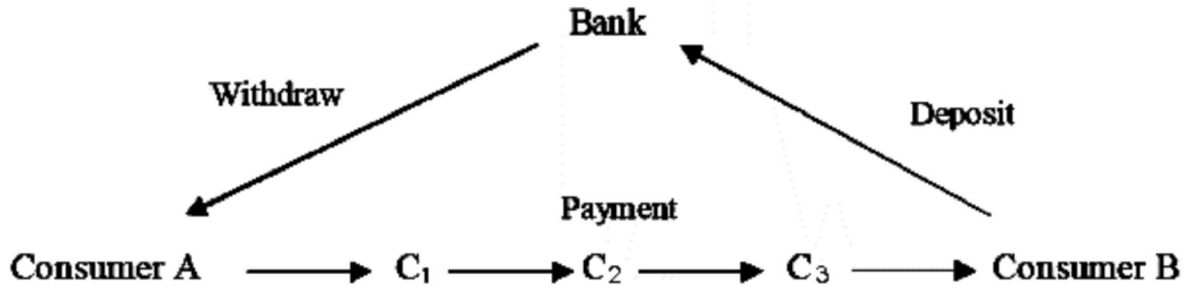
Privacy in E-cash means the existence of anonymity for the consumers who made the payment. Similar to coins and paper notes there should not be any link or trace to individual who uses the E-cash for any transaction. This feature is needed in order to protect consumers' privacy from being monitored for the purpose of financial surveillance. However, anonymity does impose certain danger such as counterfeiting, money laundering and blackmailing. Consumers should be aware that the more anonymity offered the less security achieved by the E-cash.

3. Portability

E-cash should be portable, similar to the conventional money where it does not depends on physical location. E-cash should be transferable via network to portable storage devices.

4. Transferability

Transferability features allow consumers to transfer E-cash from one person to another without a need to refer to the bank. Similar to conventional cash where coins or paper notes can be transferred easily, E-cash should be able to do the same. However, this feature imposes problem where double spending could not be trace since it might have been transferred to different entities too many times. The below diagram illustrates the transferability process of E-cash.



5. Divisibility

By divisible, it means E-cash should possess the ability to make change where E-cash can be divided into small denominations to allow small value transaction possible (this is known as micropayment). The challenge for divisible system is to be able to divide the E-cash value to small values where the total of the small E-cash value is equal to the original value. There are many systems being developed to solve divisible problem such as proposed by Eng, and Okamoto's scheme, Okamoto's scheme and Okamoto and Ohta's scheme, to name the few.

Q14. Explain the metrology of E-cash implementation.

Ans :

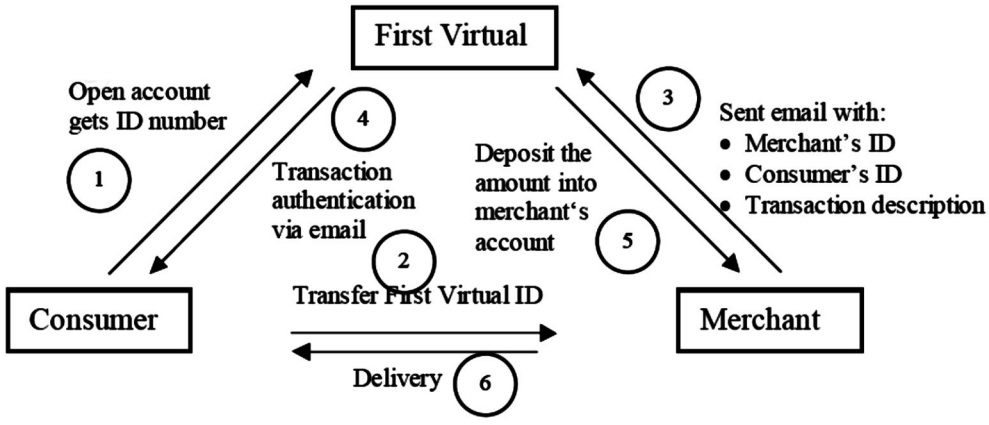
This section discusses some of the famous implementations of E-cash. Some companies presented might no long in operation or change name, the objective of the paper is get the understanding of E-cash implementation. The discussion will be focused on the methodology used and the properties presented earlier.

I) First Virtual

First Virtual Holdings founded by Lee Stein in the late 1994 is one of the first companies who offer E-cash transferable via Internet. The system depends on electronic mail as the meant of communication among consumers, merchants and First Virtual. Consumers have to give away their credit card numbers as an exchange to First Virtual E-cash. Consumers and merchants will be charge with extra charges for billing process. In summary, the First Virtual E-cash system works as follow:

1. Consumer opens account with First Virtual. Consumer must have an email account and credit card. First Virtual will give consumer an ID number as an exchange to consumer credit card number.
2. When consumer wants to purchase something from merchant who accepts First Virtual ID numbers, consumer will negotiate the price with merchant. Once agreed, consumer will give the merchant his or her First Virtual ID number.
3. The merchant then send an email to First Virtual's Internet Payment System server together with merchant's ID, consumer's IDs and description of the transaction such as the agreed price.
4. Upon receiving the merchant's email, the payment server will send an email to consumer for confirmation.
5. Consumers must reply to the email with any of these three answers:
   - YES, means consumer agrees with the transaction and allows First Virtual to instruct the bank to debit the stated amount from consumer's credit card.
   - NO, means consumer disagrees with the transaction and therefore no payment will be made. First Virtual however, will record all the refused transactions. This is done in order to avoid consumers from taking advantage of the merchants. Consumers who refuse transactions too often will then face the possibility of account termination.
   - FRAUD means consumer do not initiate the transaction. First Virtual will conduct an investigation to determine the truth.
6. Once First Virtual acknowledges that the consumer has paid the credit card company, First Virtual will credit the amount to merchant's account.

Below diagram illustrates the flow of First Virtual system.



From the description given above, First Virtual system can be categorized as identified on-line implementation where every transaction is being recorded and traceable with the need of a third party for verification. It also means the system does not provide privacy to consumers. In addition to consumer-to-merchant transaction, First Virtual also offers person-to-person E-cash transfer; therefore the E-cash

introduced is transferable. This system has shown that the used of email makes it more portable since consumers could make the transaction anytime and anywhere, as long as there is a place for accessing email. Since the Internet infrastructure is getting better by the hour, consumers should not have any problem accessing email to initiate or verify transactions.

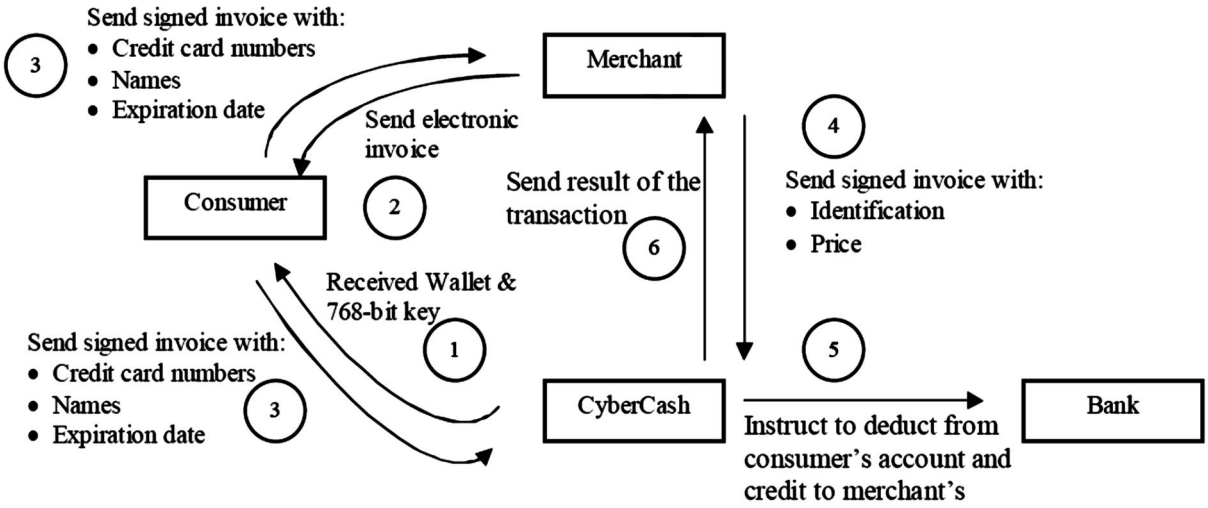
However, this system does not use neither encryption nor digital signature when sending email from consumers-to-merchants, merchants-to-First Virtual, First Virtual-toconsumers and vice versa. Although the system claims that the security achieved by not having credit card numbers transfer on the Internet but the transfer of First Virtual ID from consumers to merchants is not secured and can be intercepted. The system also emphasize that consumers verification via email is enough to secure the transaction, but then again the email message is transfer on the open network in the plain text where the email can be intercepted and sabotaged by others. Even though First Virtual implementation does not employs encryption, it is possible for the parties involve to secured their emails and their transactions. Meaning, consumers can encrypt their emails (could use non-commercial PGP) before submitting, merchants can development secure communication applications for consumers by utilizing secure protocol such as SSL to transfer the First Virtual ID and merchants can also use secured email to send details to the banks.

II) CyberCash

CyberCash is an U.S.A based company, founded by Bill Melton and Daniel Lynch in 1994. CyberCash system is using what they called "Wallet" as a medium to handle credit cards, currencies, checks and CyberCoin. CyberCoin is a system to handle micropayment less than $10. CyberCash supports not only consumer-to-merchant but also consumer-toconsumer services. Below is the description of how the implementation works:

1. Consumer request "Wallet" by downloading the free software from CyberCash Internet server. The software will establish links among consumer, merchant, CyberCash and consumer's bank. Consumer then will received a 768-bit RSA key and use a password to secure the key.
2. Once consumer decides to purchase, he or she will sends his or her "Wallet" via the communication software by pressing the "PAY" button. The system will then activate merchant's CyberCash software on merchant's storefront. The merchant sends consumer an electronic invoice with the detail information of the transaction.
3. Upon receiving the invoice, consumer will sign the invoice by adding his or her credit card number, name as appeared on the card and the credit card expiration date. The "Wallet" will encrypt the signed document with CyberCash's public key and send the document to both CyberCash and merchant.
4. Merchant who received the signed document will then add merchant's identification information and price before signing it and forward it to CyberCash.
5. CyberCash who received signed documents from both consumer and merchant will unblind the document and compare the stated price. If the stated price is the same, CyberCash will instruct the bank to deduct the agree amount from consumer's credit card, credit the same amount to merchant's account. Details of the transaction are then send to the merchant.
6. Merchant will finally deliver the purchased product or service.

The below diagram summarizes CyberCash processes:



The implementation of CyberCash is based on on-line identified E-cash. Every transaction is recorded, traceable and needs third party verification. The use of encryption enables the documents to be authenticated. The system does support divisible property with the introduction of CyberCoin. However, CyberCash does not protect consumer's privacy where consumer and merchant's identities are revealed before any transaction can be completed. In term of portability, "Wallet" can only be installed on consumer's computer; therefore consumer could only make transactions from a computer where the "Wallet" is installed. This system support both transfers but from consumer-to-consumer and consumer-to-merchant transfers.

III) DigiCash

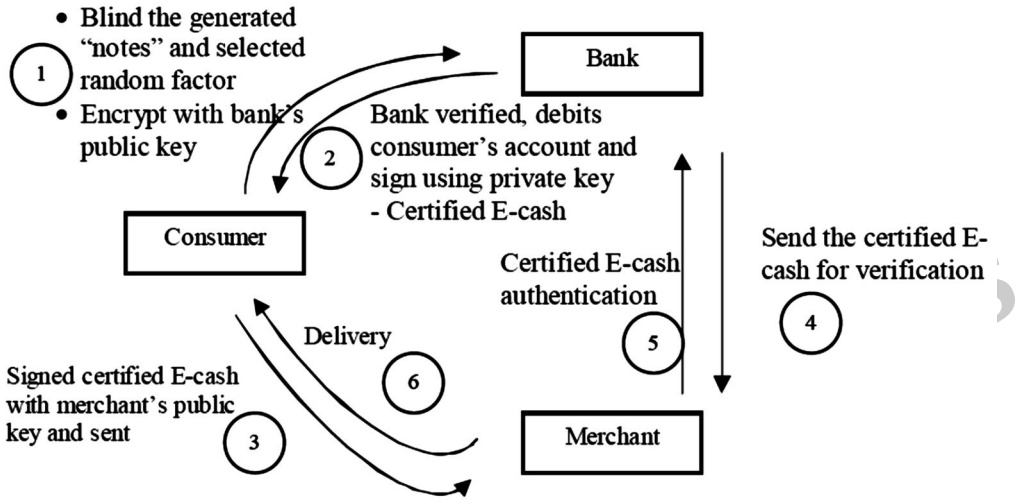
Founded by David Chaum in 1994, Digicash is located in Amsterdam. The system was designed based on Chaum's digital cash system. DigiCash system uses digital signature for encryption and "blind" signature for authentication to ensure the security of transactions and to protect consumers, merchants and banks from illegal activities. DigiCash was designed to provide payment from one computer to another computer through Internet. DigiCash offers both anonymity and identified for both of its on-line and off-line services. The E-cash product introduced by DigiCash is called "ecash" where it uses RSA encryption algorithm. The system works as follow:

1. Consumer who wants to use DigiCash must open an account with bank that provides on-line DigiCash system.
2. Ecash software will generate a pair of keys, private and public keys when it is first executed on consumer's computer. The consumer will keep the private key, which is use to sign E-cash transactions originated from consumer. Public key will be made available for banks, merchants and other people to verify any messages or E-cash transferred from consumer.
3. When consumer decides to purchase, consumer's computer will determine the denominations based on the amount needed and generate matching random serial numbers acting as "notes" for each denomination. The consumer's computer will also generate a selected random factor use to blind the denominations/random serial numbers. The blinded random serial numbers or the blinded "notes" are then encrypted with the bank's public key before sending it to bank for certifying.
4. The bank decrypts the message using it's private key. Once the message is decrypted, the bank will debit the amount found from the message from consumer's account. In exchange with the debited

amount, the bank then certified the blinded "notes" found in the message with it's private key. The signed blinded "notes" is then send back to consumer who will take out the blinding factor before using the "notes" in payment. Both the random serial numbers (the "notes") and the signatures (consumer's and bank's) are the certified E-cash.

5. Upon payment, the certified E-cash is sent to merchant who will send it to the bank for verification. The bank will verify whether E-cash is valid and have not been used before.

The diagram shown below is the visualization of the above process:



DigiCash system offers both anonymity for on-line and off-line services and the system allows transfers from consumer-to-consumer in additional to consumer-to-merchant. The certified E-cash is portable since it is a softcopy based, where it can be stored and transferred to other devices, making it easy and convenient to use. It also protects consumer's privacy via blind signature. The bank cannot make any connection as to who signed the document because only consumer know the random factor use in the blind signature. Another property is the divisible feature that comes from the introduction of CyberCoin to handle micropayment. In term of security, DigiCash provides better security by using digital signature to authenticate the message send and received. Using this approach, bank's public key is available for both consumer and merchant, making it possible for both parties to authenticate the message. However, both parties are unable to forge bank's signature since only the bank has the matching private key to sign (certified) the E-cash. Consumer is also being protected against illegal merchant activities and mistreated attempts by bank.

IV) Mondex

The development of Mondex started in the early 1990s. The concern on security has brought Mondex (E-cash application) and Multos (E-cash smart card based operating system) to the highest achievement in security recognition; level E6 was awarded in 1999 by UK IT security Evaluation and Certification (ITSEC). Recognition has proved that Mondex is one of the most secured E-cash applications available today. Mondex is an Ecash application, based on smart card where the E-cash is stored in the chip located in the smart card.

The concept of Mondex is similar to DigiCash. Consumer requests E-cash from bank. When consumer decides to purchase, consumer's E-cash will be transferred to the merchant who will then send it to the bank for verification and cashing. Upon receiving the E-cash, bank will verify and certify the E-cash, at the same time consumer's accounts will be debited and the same amount will be credited to the merchant's account. Finally, the merchant will deliver the products or services to the consumer. Mondex is offering

anonymity on both of its on-line and off-line services. For off-line transactions, merchant can do verification after the transaction completed (This might expose merchant to double spending that is difficult to trace). Besides consumer-to-merchant transaction, Mondex also allows consumer-to-consumer E-cash transfer. In short Mondex had fullfilled almost all the desirable properties of E-cash mentioned earlier. It has security, which is based on digital signature where each message transfer among bank, merchantand consumer can be authenticated. The system is portable with the use of smart card. Mondex system also protects consumer's privacy by using blind signature. In term of divisibility, Mondex declares that the system is able to handle micropayment as small as one cent.

5.6 Secure Electronic Transaction (SET)

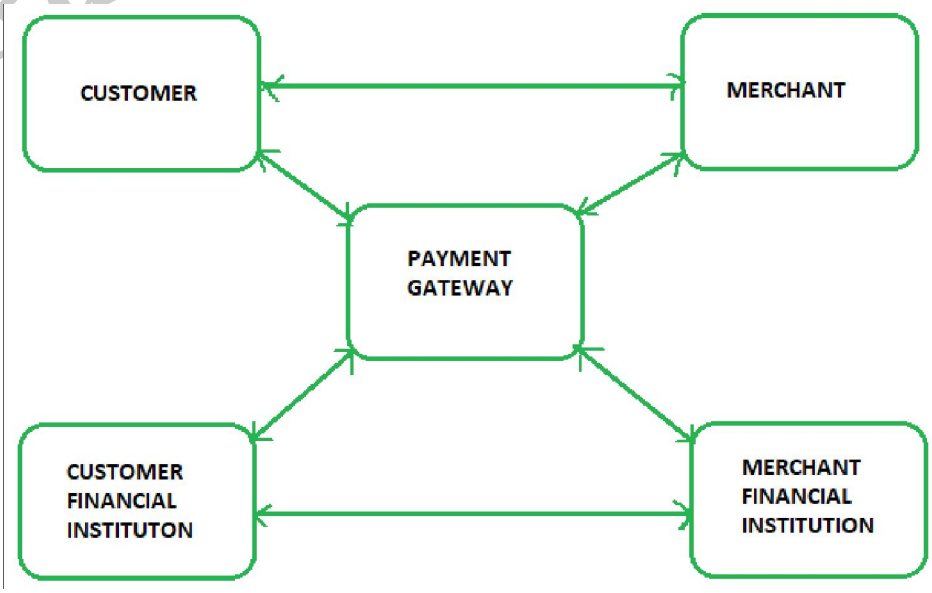
Q15. Explain the concept of Secure Electronic Transaction (SET).

Ans : **(Imp.)**

Meaning

Secure Electronic Transaction or SET is a system which ensures security and integrity of electronic transactions done using credit cards in a scenario. SET is not some system that enables payment but it is a security protocol applied on those payments. It uses different encryption and hashing techniques to secure payments over internet done through credit cards. SET protocol was supported in development by major organizations like Visa, Mastercard, Microsoft which provided its Secure Transaction Technology (STT) and NetScape which provided technology of Secure Socket Layer (SSL). SET protocol restricts revealing of credit card details to merchants thus keeping hackers and thieves at bay. SET protocol includes Certification Authorities for making use of standard Digital Certificates like X.509 Certificate.

Before discussing SET further, let's see a general scenario of electronic transaction, which includes client, payment gateway, client financial institution, merchant and merchant financial institution.



Requirements in SET

SET protocol has some requirements to meet, some of the important requirements are :

- It has to provide mutual authentication i.e., customer (or cardholder) authentication by confirming if the customer is intended user or not and merchant authentication.
- It has to keep the PI (Payment Information) and OI (Order Information) confidential by appropriate encryptions.
- It has to be resistive against message modifications i.e., no changes should be allowed in the content being transmitted.
- SET also needs to provide interoperability and make use of best security mechanisms.

Participants in SET

In the general scenario of online transaction, SET includes similar participants:

1. **Cardholder:** customer
2. **Issuer:** customer financial institution
3. **Merchant**
4. **Acquire:** Merchant financial
5. **Certificate authority:** Authority which follows certain standards and issues certificates (like X.509V3) to all other participants.

SET functionalities :

- **Provide Authentication**
- **Merchant Authentication** – To prevent theft, SET allows customers to check previous relationships between merchant and financial institution. Standard X.509V3 certificates are used for this verification.
- **Customer / Cardholder Authentication** – SET checks if use of credit card is done by an authorized user or not using X.509V3 certificates.
- **Provide Message Confidentiality** : Confidentiality refers to preventing unintended people from reading the message being transferred. SET implements confidentiality by using encryption techniques. Traditionally DES is used for encryption purpose.
- **Provide Message Integrity** : SET doesn't allow message modification with the help of signatures. Messages are protected against unauthorized modification using RSA digital signatures with SHA-1 and some using HMAC with SHA-1.

5.7 MICRO PAYMENTS

Q16. Explain about micro payments.

(OR)

Describe the concept of mixro payments.

Ans : **(Imp.)**

Micropayment system provides a means of transferring small monetary amounts and serve as a convenient alternative to traditional payment arrangements. Micropayments refer to low value electronic transactions.

MicroPayment Systems (MPS) have been developed in the past decade, but for various reasons, have not yet penetrated the market deeply. The critical mass of users will probably be reached very soon in some industrial countries where Internet has a deep penetration. This has an effect in the fragile balance between the defenders and the offenders of computing systems. There is a need to analyze this new player and understand the new issues that arise by the introduction of MPSs to our everyday life.

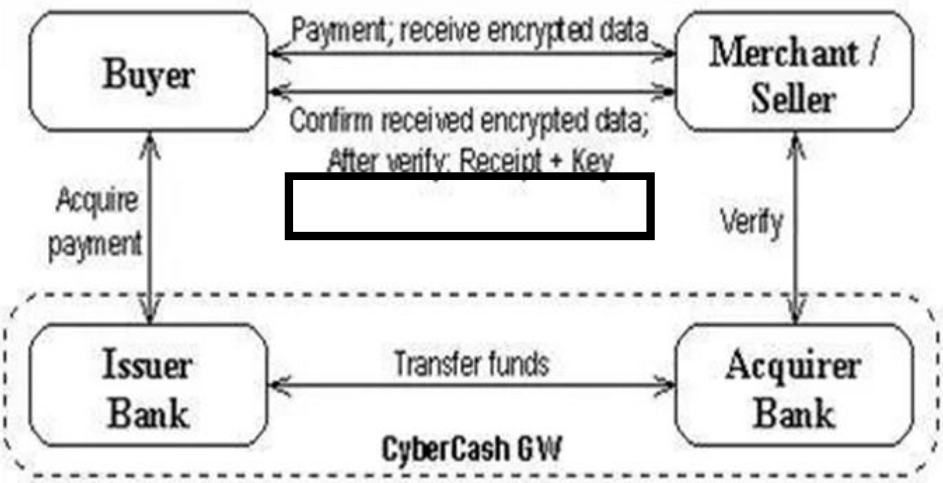
Four digital money systems were chosen. These are:

1. Cyber Cashs Cyber Coin
2. Digi Cashs eCash
3. Digitals Millicent
4. Meteore

1. Cyber Cashs Cyber Coin

CyberCash is an electronic credit card scheme CyberCoin services are distributed through banks, which offers online merchants the CyberCoin service and offers consumers co-branded "Wallets."

To be able to purchase an item, the buyer first sends a payment order to the merchant who forwards it to CyberCash Gateway Server for verifying. Both buyers and merchants have an electronic wallet of their own. The wallets are required for bookkeeping the financial information of the parties.



The flow of Money of CyberCashs CyberCoin method is shown in the figure(a) above are,

1. To be able to use CyberCoin system the buyer must have executed an off-line withdrawal of electronic money onto the wallet.
2. The buyer wishes to initiate a purchase by contacting the merchant.
3. The merchant receives this request and sends the buyer the electronic merchandise encrypted with DES (Data Encryption Standard – a block cipher developed by IBM in mid 70's), without the corresponding DES key.
4. The buyer confirms the purchase and acknowledges that he has received the purchase request data by sending back the notational order which contains the financial information about buyer's account. This transferred data must be hidden from the viewpoint of the merchant.
5. The merchant sends this payment order to Cyber Cash for verifying purposes.

6. Then Cyber Cash sends its approval to the merchant if the payment order is valid. It should be noted that Cyber Cash need not contact the actual bank for this transaction because Cyber Cash maintains the equivalents of the accounts of both the buyer and the merchant. The buyer's and the merchant's wallet balances are adjusted according to the payment value of the purchase.
7. Finally, the merchant can send the DES key to the buyer who can decrypt the previously received encrypted digital goods.

2. DigiCashs eCash

This system is based on what is called a single user token system. This system often called as in-line System where the authentication of users is done.

The flow of Money of Digicashs eCash method is shown in the following steps as,

1. The user generates blinded electronic bank notes and sends them to his bank to be signed with his bank's public key (PK).
2. The bank signs the notes, deducts the amount from the user's account, and sends the signed notes back to the user.
3. The user removes the blinding factor and uses them to purchase at the shop.
4. The shop verifies the authenticity of the bank notes using the bank's corresponding public key and sends them to the bank where they are checked against a list of notes already spent.
5. The amount is deposited into the shop's account, the deposit confirmed, and the shop in turn sends out the goods.
6. All communication over the network is protected by encryption.

3. DECs Millicent

Millicent is a decentralized micro-payment scheme, which is designed to allow payments as low as 1/10 of a cent. It uses a form of electronic currency, which is called scrip .

Scrip is an electronic manufacturer coupon that has a pre-paid value, just like the conventional cash does. It has an intrinsic value, but it is valid only with a specific vendor. In other words, it can be spent only in the context to which it was originally meant.

DECs, It is designed to make the cost of committing a fraud, more than the value of the actual transaction. It uses symmetric encryption for all data transactions. Millicent transactions are not anonymous and mostly off-line.

Millicent is perhaps the smallest and simplest implementation of the contemporary systems. The mechanism is basically a very simple protocol for two participants which is extended in the realm of systems to cover a multitude of payment situations. It is, however, designed for the handling of a series of inexpensive and casual transactions.

The principal actors of the scheme are the Broker, the Customer and the Vendor. Figure (b) demonstrates the scheme.

i) The Broker

The Broker mediates between Vendors and Customers in order to simplify the tasks they perform. He acts like a bank and provides the electronic currency (scrip) for the micro-payments. A Broker, after coming to a deal with the Vendor, can either generate his own valid Vendor-specific scrip, or buy a large amount of scrip, from the Vendor, using a macro-payment system. The Broker is then selling the scrip to the customers via micropayment transactions. During Customer purchases (either from Broker or Vendor), no transactions between the Broker and the Vendors are taking place

ii) The Customer

The Customers buy scrip from the Brokers, using real money, via a macro payment system. The amount should be sufficient to cover the transaction cost plus to produce financial gain for both the Broker and the Vendor (scrip is Vendor specific). The Customer can then use the scrip to perform micro-payment purchases. No transactions with real money are taking place in any given time between Customers and Vendors.

iii) The Vendor

The Vendor is the data bank. He supplies customers with data, services or both. He accepts his specific scrip as the only method of payment. The scrip was either generated by him (the Vendor) or by a licensed broker. After that the Vendor can transmit the requested data back to the Customer, using a given encryption algorithm for avoiding fraudulent use.

5.8 CASE STUDIES OF ENTERPRISE SECURITY (.NET AND J2EE)

Q17. Explain the elements of .NET?

(OR)

What are the elements of .NET?

Ans : **(Imp.)**

The .Net initiative, a significant undertaking by Microsoft, is an attempt to tie together applications, development languages, operating systems and data stores into a unified, distributed enterprise platform.

The .Net Framework is a reworking of Windows DNA, Microsoft's previous enterprise development platform, consisting of the CLR (Common Language Runtime), base classes and presentation through ASP.Net. .Net is tightly integrated into the Windows OS environment (with .Net support integrated into all of Microsoft's enterprise applications), but the portability presented by the CLR/CLS (Common Language Specification)/CTS (Common Type Specification) combination means that if Microsoft wanted, .Net could be implemented on other platforms.

i) C#

C# is an object-oriented language, derived from C++, and syntactically similar (with some exceptions) to the Java language. C# is a "brand new" language, and may suffer from teething troubles as a result. While it is not a part of the .Net framework, it is a part of the .Net strategy, and mirrors Java's function within J2EE.

ii) Common Language Runtime (CLR)

All .Net components are compiled into an interim language called Intermediate Language (IL), which is then executed in the runtime environment provided by the CLR. A growing number of compilers for languages such as C#, Visual Basic. Net, C++, etc. are available for use with .Net, giving programmers the ability to work in .Net using familiar languages. The CLR is part of the core of the .Net Framework, providing a secure execution environment through the principles of what Microsoft describes as managed code and code access security. The CLR also provides housekeeping functionality through garbage collection.

iii) ASP. Net

ASP.Net is an onward progression from ASP (Active Server Pages). Differences from ASP include programmatical changes and the fact that ASP.Net pages are now compiled and executed in a CLR.

iv) ADO.Net

ADO.Net is the successor to Microsoft's ActiveX Data Objects (ADO). Whereas ADO

provided two-dimensional access to data, through the use of rows and columns, ADO.Net describes data as objects and utilises XML for transmitting data between application tiers.

v) Assemblies

An assembly is a collection of code used as the building block of a .Net framework deployment. Assemblies are designed to include security information (in terms of permissions requested and strong name), versioning information (which enables multiple versions of software to co-exist on a single system, theoretically eliminating DLL conflicts), code and supporting resources. Assemblies can be built using the Visual Studio. Net IDE and the Assembly Generation Tool.

Q18. Explain the architecture of .NET.

Ans : **(Imp.)**

The CLR and base classes are responsible in large part for security functions within the .Net framework. The CLR uses a number of criteria to determine the security permissions of code to be executed, and obtains some security information from XML configuration files.

The base classes control access to resources such as the file system by determining the permissions of the caller.

.Net's two primary security functions are described in the following sections.

1. Code access security

Code access security is a core part of the CLR's security function. Code access security determines the level of trust assigned to a piece of code based on its origins, publisher and other factors. Some key concepts used to define code access security are described below.

i) Evidence-based security

.Net uses the concept of "Evidence-based security", to describe the process by which assemblies are examined by the CLR at runtime. The CLR queries assemblies using the following primary criteria:

- Where did this code originate?
- Who created this assembly?

The first question can be broken out into "Which URL did the assembly originate from?" and "Which zone did the assembly originate from?". Microsoft uses the concept of zones to describe security environments like the Internet, local networks or intranets, the local machine, etc.

Evidence can also be in the form of a digital certificate signed by the publisher of the assembly or the strong name (a unique identifier, consisting of a text name, digital signature, and a public key) of the assembly.

The second question is reasonably straightforward – "What information is available on the creator of this assembly?".

The evidence gathered is included as part of an assembly's metadata. Metadata can include information on types, relationships with other assemblies, security permissions requested as well as a description of the assembly. Metadata is examined by the CLR at various points, including by the verifier, which ensures that types are correct prior to compilation of IL to native code. The verifier is also responsible for ensuring that the metadata associated with an assembly is valid.

The CLR examines the metadata to establish an identity for the assembly and determines the permissions allocated to the assembly based on the security policy.

ii) Permissions

Permissions within the CLR are the building blocks of access control. A code access permission in this context is the right of a piece

of code to access a particular resource or perform a particular operation. When assemblies are built, the permissions that it requires to run can be included as part of its description.

At runtime the assembly requests these permissions, and those permissions that it requests are either granted or denied by the CLR. An assembly will never be given greater permissions than the current security policy allows, but may be given lesser permissions than it requests.

Examples of permissions are Security Permission (the permission to view or modify the security policy), UIPermission (the right to create sub-windows and make use of the clipboard) and the RegistryPermission class (which gives access to Windows Registry details).

Permissions can be grouped into permission sets for ease of administration. These permission sets are then associated with code groups, which are described in the next section. Examples of permission sets are Nothing (no permissions at all, so code cannot execute), Internet (the permissions assigned to untrusted code) and Everything (the set of permissions that grants all standard permissions, with the exception of avoiding code verification).

At runtime, the CLR performs what is known as a stack walk to determine whether the calling assembly has permission to access a particular resource, checking requested permissions against granted permissions for each caller on the stack.

If an assembly's request for access to a resource is denied, a SecurityException is thrown.

iii) Security Policy

The security policy is administered using the Code Access Security Policy Tool (Caspol.exe) or the .Net Framework configuration tool. Administrators set the policy for assemblies and application domains, the CLR uses the evidence described above to identify an assembly, and then uses the security policy to determine the permissions the assembly has at runtime.

The policy identifies code by organising code into groups that categorise based on the evidentiary information described, such as the zone from which the code is loaded. If no other information is available, the default policy for the zone from which the code was obtained will be used to determine the permissions that an assembly has.

2. Role-based security

User membership in a role within a .Net application helps to determine the access that the user has to perform particular operations and access resources. For example, in the case of a financial application, a Broker role might have permission to initiate, authorise and cancel trades on behalf of an individual or financial institution.

Role-based security describes the means by which the .Net framework identifies, authenticates and authorises an individual user from any of a number of authoritative sources. When a user is identified, their authenticated identity (and role membership) is denoted by the term principal. A principal can be a member of one or more roles; for example, an individual could be a member of a Broker and an Executive role.

Role-based security employs a permission structure similar to that of code access security, with permissions managed through the Principal Permission object.

Authentication and authorisation sources for users can be through Windows machine security, domain security or some custom authentication source.

3. Programmatic security

.Net uses the IsInRole method to determine whether a user belongs to a particular role. For example, to determine if a user has membership of a Broker role, the following could be used :

```
If User.IsInRole("Broker")
' Permit requested function
Else
' Bounce back to login
End If
```

The classes responsible for the determination of principal identity are stored in the System.Security.Principal namespace.

Q19. What are the core elements of J2EE.

Ans :

J2EE is Sun's reference standard for enterprise development, first introduced in December 1999, and now at version 1.3. Core elements of J2EE are described in this section.

i) Java language

Java is an object-oriented language derived from C++, with features that simplify coding such as memory management through garbage collection, no pointers, etc. Java is designed to be "Write once, run anywhere", this goal being achieved through the use of portable bytecode.

ii) JVM/JRE

The JRE (Java Runtime Environment) consists of the Java Virtual Machine, a just-in-time compiler and some foundation classes. Java classes are compiled into platform-independent bytecode that is executed in the JVM.

iii) JSPs and Servlets

Java Server Pages (JSPs) are analogous to ASP technology, providing the capability to build dynamic web pages composed of HTML with embedded dynamic components, e.g. references to Beans.

Servlets are described as applets that run on the web server, and are used where CGI would traditionally have been employed to build dynamic web applications.

iv) EJB

Enterprise Java Beans (EJB) are used to build distributed applications by providing the communications and execution framework for distributed components. Critical services provided by an EJB container include transaction management, security, resource management and persistence.

v) JDBC

Java Database Connectivity (JDBC) is an API for connecting to relational databases, manipulating their contents, and processing the output of issued SQL statements. Numerous database vendors have developed drivers based on the JDBC API.

Q20. Explain the architecture of J2EE.

Ans : **(Imp.)**

1. Code management through the JVM and the class file verifier, the class loader and the Security Manager

Basic Java security employs the concept of a "sandbox" to limit the abilities of untrusted code to cause damage to the system on which it runs. Historically, an untrusted piece of code such as an applet would be disallowed from accessing local disk, opening network connections, etc. With the introduction of certificate support through the Java Plug-In, the origin and author of a signed applet could be established definitively, enabling fine-grained permissions to be assigned to individual applets based on the security policy. This means that applets are no longer confined to the default sandbox.

As described in the coming paragraphs, the sandbox is implemented through the JVM and its class verifier, but also through the class loader and the Security Manager/ACL manager.

i) JVM security

The JVM provides a secure runtime environment by managing memory, providing isolation between executing components in different namespaces, array bounds checking, etc. The dynamic way in which the JVM allocates the various memory areas (method area, GC heap, thread stacks) means that it is almost impossible for a would-be attacker to determine what memory areas to attempt to insert malicious instructions into. Bounds checking on arrays prevent unreferenced memory accesses.

The JVM's class file verifier examines classes for basic class file structure upon loading. While bytecodes produced by Sun's compiler should be relatively free of errors (type errors, for example), it is possible for an attacker to manually create malicious bytecode.

The class file verifier examines each loaded class file in four passes; from the most basic check, where the physical attributes of the file are checked (size, magic number, length of attributes) to checking the constant pool to ensure that method and field references have correct attributes, to parsing the instructions for each method. Note that, by default, the only trusted classes are the base classes. All other classes, including those loaded from the application classpath are considered untrusted and must be verified.

ii) The Class Loader architecture

There are two types of class loader – the primordial class loader, which is a part of the JVM, and class loader objects, which are used to load non-essential classes.

There can only be one primordial class loader, which is used to bootstrap the JVM by loading the base classes, sometimes using native OS-dependent means.

There can be many instances of class loader objects in the JVM, which can be instantiated on the fly, and used to load objects from sources such as the network, local disk, or data stores. Controlling the creation of class loader objects is therefore important due to the function of the class loader.

Class loaders are responsible for locating classes requested by the JVM for loading into the runtime environment. Part of their responsibility is to prevent unauthorised or untrusted code from replacing trusted code that makes up the base classes. As an example, a class loader should prevent the replacement of the Security Manager class. The attempted replacement of a base class by a maliciously coded class is referred to as class spoofing.

All class loaders, with the exception of the primordial class loader, are loaded by other class loaders, which become the loaded class loader's parent. Thus, the loading of class loaders describes a tree structure with the primordial class loader at the root and classes as the leaf nodes (obviously class loaders are themselves classes).

If a class loader loads a class, all subsequent requests for related classes are directed through that class loader. For example, if a class loader "A" loads a class "Building", and "Building" makes calls to methods in a class called "Cubicle", the JVM will use class loader "A" to load "Cubicle" and any other classes that are referred to by "Building".

Class loaders prevent class spoofing by passing requests back up the tree through their parent class loader until the class loader that loaded

the requested class is reached. In the case of the Security Manager class, the primordial class loader is responsible, and consequently the malicious class described above will not be loaded. Classes are loaded only once, and the base classes are only loaded from the local disk using the system classpath.

Class loaders also provide security by managing namespaces. A class that is loaded by a particular class loader can only reference other classes in the same namespace, i.e., loaded by the same class loader or its parents.

iii) The Security Manager and Access Controller

The Security Manager was responsible for examining and implementing the security policy, which is specified by policy files. The security policy, as with .Net, determines the permissions that code has at runtime.

In more recent versions of Java, the decisions on whether to grant permissions based on security policy are delegated to the Access Controller. When a class makes a request for permissions, it is received by the Security Manager which passes the request to the Access Controller.

In a similar fashion to the way that .Net groups permissions into permission sets, which it then associates with code groups, permissions in the Java world are grouped into protection domains, associated with code sources. In other words, groups of permissions are associated with groups of classes, the classes being grouped by their origin.

Signed Java code is assigned permissions based on the system policy as applied to the code's protection domain. Depending on the permissions associated with the source of the code, the applet may have full access to system resources, or may be restricted to a small subset.

Java applications, by default, have no associated security manager, and therefore have full access to system resources.

2. Platform Roles

The J2EE platform specification describes Organisational or Platform Roles that can be used to divide up responsibilities in a J2EE development and deployment cycle. The Roles described in the platform specification are Product Provider, Application Component Provider, Application Assembler, Deployer and Systems Administrator. These Roles are not absolutes – the responsibilities of the various roles could be divided differently to fit a company's development and deployment methodologies.

Of these roles, most have clear security implications. The Product Provider and Application Component Provider roles are responsible for developing secure code, the Assembler is responsible for defining method permissions and security roles, the Deployer verifies the security view of the deployed application and assigns principals to roles, and the Systems Administrator administers principals and ensures that the local security environment is correct for the J2EE platform.

3. Security Roles and the Deployment Descriptor

The deployment descriptor is an XML file that ships with each EJB[1], and describes declaratively many of the aspects of the EJB's function and makeup, and its relationship with other beans. One of the elements in the descriptor is the <security-role-ref> element. This element type is used by the bean developer to define all of the security roles used in the EJB code.

Security role names are associated with links, which are then called elsewhere in the descriptor.

The <security-role> element is used to call roles described in the <security-role-ref> elements. For example:

.

.

.

<security-role-ref>

<role-name>root</role-name>

<role-link>super-user</role-link>

</security-role-ref>

.

.

<security-role>

<description>This is the security-role for the role "root", defined above</description>

<role-name>super-user</role-name>

</security-role>

The description field in the <security-role> descriptor element is optional.

Membership of a role confers a set of permissions for the duration of the role membership. Principals can be in several roles at the same time, e.g. employee and manager.

Method permissions are also described in the deployment descriptor.

Programmatic security

Role membership can be determined programmatically in the J2EE environment using the is User In Role and get User Principal methods of the HTTP Servlet Request object for the web container.

As part of the bean-container contract, the container provides the EJB Context object. The corresponding EJB Context methods are is Caller In Role and get Caller Principal. get Caller Principal returns the principal associated with the security context, while, predictably enough, is Caller In Role is a boolean method used to determine whether the caller belongs to a specified security role.

FACULTY OF INFORMATICS

M.C.A II Year III - Semester Examination

MODEL PAPER - I

NETWORK SECURITY

Time : 3 Hours] [Max. Marks : 70

Note: Answer all the question according to the internal choice **(5 × 14 = 70)**

ANSWERS

1. Explain General Threats to Computer Network. **(Unit-I, Q.No. 10)**

OR

2. Describe the model for network security. **(Unit-I, Q.No. 5)**

3. (a) Describe the concept of DES algorithm. **(Unit-II, Q.No. 1)**

   (b) Explain triple DES with two keys. **(Unit-II, Q.No. 4)**

OR

4. Explain about the applications of AES. **(Unit-II, Q.No.10)**

5. Explain the concept of Hash Function. **(Unit-III, Q.No.1)**

OR

6. Explain about Ciphes Message Authentication code(CMAC). **(Unit-III, Q.No.8)**

7. Explain the concept of Digital Certi- ficate and CA. **(Unit-IV, Q.No. 2)**

OR

8. Explain the concept of firewall configuration. **(Unit-IV, Q.No.8)**

9. Explain the mechanism of kerberos ticket. **(Unit-V, Q.No.4)**

OR

10. Explain the concept of IPSec. **(Unit-V, Q.No.10)**

FACULTY OF INFORMATICS

M.C.A II Year III - Semester Examination

MODEL PAPER - II

NETWORK SECURITY

Time : 3 Hours] [Max. Marks : 70

Note: Answer all the question according to the internal choice **(5 × 14 = 70)**

ANSWERS

1. Explain different types of security attacks. **(Unit-I, Q.No. 4)**

OR

2. (a) What is IP Spoofing? **(Unit-I, Q.No. 7)**

   (b) What is replay attack? **(Unit-I, Q.No.8)**

3. (a) Explain the strength of DES. **(Unit-II, Q.No. 3)**

   (b) Briefly explain about Diffie - Hellman algorithm. **(Unit-II, Q.No. 15)**

OR

4. Explain various principles of public key cryptography. **(Unit-II, Q.No.12)**

5. Describe the concept of message degest algorithm. **(Unit-III, Q.No.2)**

OR

6. What are the requirements for message authentication codes? Explain briefly. **(Unit-III, Q.No.6)**

7. What is VPN. State its benefits. **(Unit-IV, Q.No. 9)**

OR

8. Explain different types of attacks on Smart cards. **(Unit-IV, Q.No.17)**

9. Explain the concept of Secure Electronic Transaction (SET). **(Unit-V, Q.No.15)**

OR

10. Explain the architecture of NET. **(Unit-V, Q.No.18)**

FACULTY OF INFORMATICS

M.C.A II Year III - Semester Examination

MODEL PAPER - III

NETWORK SECURITY

Time : 3 Hours] [Max. Marks : 70

Note: Answer all the question according to the internal choice **(5 × 14 = 70)**

ANSWERS

1. What are the various attributes of security? **(Unit-I, Q.No. 3)**

OR

2. (a) Explain the concept of Man-in-the-Middle attacks. **(Unit-I, Q.No. 9)**

   (b) What is Denial of service attack? **(Unit-I, Q.No.6)**

3. (a) Define AES. Explain the features of AES. **(Unit-II, Q.No. 5)**

   (b) Explain about Symmetric Key distribution using Asymmetric Encryption. **(Unit-II, Q.No. 11)**

OR

4. Explain about Elliptic Curve Cryptography. **(Unit-II, Q.No.14)**

5. (a) Define Biometric Authentication. Explain its types. **(Unit-III, Q.No.12)**

   (b) Explain DSA algorithm. **(Unit-III, Q.No.11)**

OR

6. What are the requirements for message authentication codes? Explain briefly. **(Unit-III, Q.No.6)**

7. Explain different types of firewalls. **(Unit-IV, Q.No. 7)**

OR

8. Explain the best practices for PKI Management. **(Unit-IV, Q.No.3)**

9. What are the various types of tickets with kerberos V5? **(Unit-V, Q.No.6)**

OR

10. Describe the concept of mixro payments. **(Unit-V, Q.No.16)**