GUDLAVALLERU ENGINEERING COLLEGE

(An Autonomous Institute with Permanent Affiliation to JNTUK, Kakinada)

Seshadri Rao Knowledge Village, Gudlavalleru – 521 356.

Department of Computer Science and Engineering

HANDOUT

on

INFORMATION SECURITY

Vision of the Department

To be a Centre of Excellence in computer science and engineering education and training to meet the challenging needs of the industry and society

Mission of the Department

- To impart quality education through well-designed curriculum in tune with the growing software needs of the industry.
- To be a Centre of Excellence in computer science and engineering education and training to meet the challenging needs of the industry and society.
- To serve our students by inculcating in them problem solving, leadership, teamwork skills and the value of commitment to quality, ethical behavior & respect for others.
- To foster industry-academia relationship for mutual benefit and growth.

Program Educational Objectives

PEO1: Identify, analyze, formulate and solve Computer Science and Engineering problems both independently and in a team environment by using the appropriate modern tools.

PEO2: Manage software projects with significant technical, legal, ethical, social, environmental and economic considerations.

PEO3: Demonstrate commitment and progress in lifelong learning, professional development, leadership and Communicate effectively with professional clients and the public.

HANDOUT ON INFORMATION SECURITY

Class & Sem. :IV B.Tech – I Semester Year :2018-19

Branch : CSE Credits : 3

==

1. Brief History and Scope of the Subject

History:

- Since the early days of writing, heads of state and military commanders understood that it was necessary to provide some mechanism to protect the confidentiality of written correspondence and to have some means of detecting tampering.
- Julius Caesar is credited with the invention of the Caesar cipher ca. 50 B.C., which was created in order to prevent his secret messages from being read should a message fall into the wrong hands.
- World War II brought about many advancements in information security and marked the beginning of the professional field of information security.
- The end of the 20th century and early years of the 21st century saw rapid advancements in telecommunications, computing hardware and software, and data encryption. The availability of smaller, more powerful and less expensive computing equipment made electronic data processing within the reach of small business and the home user. These computers quickly became interconnected through a network generically called the Internet or World Wide Web.
- The rapid growth and widespread use of electronic data processing and electronic business conducted through the Internet, along with numerous occurrences of international terrorism, fueled the need for better methods of protecting the computers and the information they store, process and transmit. The academic disciplines of computer security, information security and information assurance emerged along with numerous professional organizations – all sharing the common goals of ensuring the security and reliability of information systems

Developments:

In recent years, a body of law has emerged that explicitly establishes certain minimum duties with regard to the security of business information systems. Outside of certain regulated industries such as telecommunications or banking, there were not formerly any laws that directly and explicitly established duties with regard

to information security. Of course, many laws can be construed as implicitly imposing a duty to maintain an appropriate level of information system security. For example, statutes that authorize commercial transactions in electronic form contain implicit information security requirements if they provide that electronic records will only be recognized as equivalent the electronic records are accurate and accessible.

In addition, it is possible to infer duties to maintain minimum levels of information

security from laws such as anti-money laundering laws, wiretap laws or securities laws. Recent information security legislation clearly departs from these earlier laws, for example by defining the subject of information security clearly or by spelling out concrete and specific duties that apply to business information systems.

2. Pre-Requisites

You need to have a decent understanding of the basics of TCP/IP. You should know the difference between IP, ICMP, TCP, and UDP. You should know what port numbers and sequence numbers are, and have (some) understanding of the TCP flags.

You should also be comfortable with sockets programming — some of the homework assignments will require you to implement network clients or servers.

3. Course Objectives:

- To familiarize with various aspects of information security.

4. Course Outcomes:

Students will be able to:

- analyze various concepts of security over networks.
- differentiate various cryptographic techniques such as conventional and modern encryption techniques.

- discuss various public key algorithms such as RSA and digital signature algorithm and special authentication techniques and various key management rules.
- apply Security mechanisms for special security for e-mails by using PGP software and Secure/MIME techniques.
- apply Security mechanisms for IP level security and web level security mechanisms.
- analyze how to give system security by using various firewalls and learn how to detect intrusion techniques.

5. Program Outcomes:

Graduates of the Computer Science and Engineering Program will have

a. An ability to apply knowledge of computing, mathematics, science and engineering fundamentals to solve complex engineering problems.

b. An ability to formulate and analyze a problem, and define the computing requirements appropriate to its solution using basic principles of mathematics, science and computer engineering.

c. An ability to design, implement, and evaluate a computer based system, process, component, or software to meet the desired needs.

d. An ability to design and conduct experiments, perform analysis and interpretation of data and provide valid conclusions.

e. An ability to use current techniques, skills, and tools necessary for computing practice.

f. An ability to understand legal, health, security and social issues in Professional Engineering practice.

g. An ability to understand the impact of professional engineering solutions on environmental context and the need for sustainable development.

h. An ability to understand the professional and ethical responsibilities of an engineer.

i. An ability to function effectively as an individual, and as a team member/ leader in accomplishing a common goal.

j. An ability to communicate effectively, make effective presentations and write and comprehend technical reports and publications.

k. An ability to learn and adopt new technologies, and use them effectively towards continued professional development throughout the life.

l. An ability to understand engineering and management principles and their application to manage projects in the software industry.

6. Mapping of Course Outcomes with Program Outcomes:

	a	b	c	d	e	f	g	h	i	j	k	l
CO			M					M				
CO	H				M						M	
CO	H					M		M			H	H
CO	M		H			M		M			M	
CO										M		
CO				M								

7. Prescribed Text Books

1. William Stallings, Cryptography and network Security, Principles and practice, Fifth edition, PHI/Pearson.

2. William Stallings, Network Security Essentials (Applications and Standards), Pearson Education.

Reference Text Books

1. Eric Maiwald , Fundamentals of Network Security, Dreamtech press.

2. Ryan Russel, Dan Kaminsky, et al., Hack Proofing your network, Wiley Dreamtech.

3. Whitman, Thomson , Principles of Information Security.

4. Buchmann, Introduction to Cryptography, Springer.

URLs and Other E-Learning Resources

URL's:

http://www.williamstallings.com/StudentSupport.html

http://eprint.iacr.org

http://www.cryptography.com

Journal: IEEE/ACM Transactions on Networking

On-Line Journal:

- ACM Transactions on Information and Systems Security http://portal.acm.org/TISSEC
- ACM Transactions on Information Systems http://portal.acm.org/TOIS
- IEEE Transactions on Information Theory http://ieeexplore.ieee.orgservlet/opac?punumber=18

8. Digital Learning Materials:

JKC CD's –2 on CRYPTOGRAPHY and NETWORK SECURITY

9. Lecture Schedule / Lesson Plan

Topic	No. of Periods	
	Theory	**Tutorial**
UNIT –1: Introduction		
Security Attacks	2	1
Security Services	1	
Security Mechanisms.	1	
A model for Internetwork security.	1	
Non Cryptographic protocol vulnerabilities	2	1
Software vulnerabilities	2	
UNIT – 2: Secret key cryptography		
Conventional encryption principles	1	2
Conventional encryption algorithms	4	
cipher block modes of operation	1	1
key distribution approaches of message authentication	1	
secure hash functions and HMAC.	2	

UNIT – 3: Public key cryptography		
Public key cryptography principles	1	1
public key cryptography algorithms	3	
digital signatures	2	
Certificate Authority and key management- Kerberos	2	1
X.509 Directory Authentication Service.	2	
UNIT – 4: Authentication applications & introduction to IP security		
Email privacy- Pretty Good Privacy (PGP)	2	1
S/MIME	2	
IP Security Architecture	2	1
Authentication Header	2	
Encapsulating Security Payload.	2	
UNIT – 5: Transport-level Security		
Web Security Requirements	2	2
Secure Socket Layer (SSL)	2	
Transport Layer Security (TLS)	2	
Secure Electronic Transaction (SET)	3	1
UNIT – 6: System security		
Intruders	1	1
Viruses and related threats	2	
Firewall Design principles	2	
Trusted Systems	2	1
Intrusion Detection Systems	2	
Total No.of Periods:	**56**	**14**

10. Seminar Topics

- Conventional encryption algorithms
- public key cryptography algorithms
- secure hash functions and HMAC.
- Email privacy- Pretty Good Privacy (PGP)
- Overview of IP Security
- SET
- Viruses and related threats

Unit – I

Objectives:

- ➢ To introduce the terms security attacks, services and mechanisms
- ➢ To know different security attacks and software vulnerabilities

Syllabus: Security Fundamentals

Security Attacks, Security Services- Confidentiality, Authentication, Integrity, Non repudiation, access Control and Availability, Security Mechanisms, A model for Internetwork security. Non Cryptographic protocol vulnerabilities- DoS, DDoS, Session hijacking and Spoofing. Software vulnerabilities- Phishing, Buffer overflow, Format String Attacks, SQL injection.

Outcomes:

Students will be able to

- ➢ differentiate security attack, service and mechanisms
- ➢ distinguish different security attacks
- ➢ summarize different security services and mechanisms
- ➢ distinguish different software vulnerabilities

SECURITY FUNDAMENTALS

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

Security Mechanism: A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.

Security Service: A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization.

1.1 Security Attack :

- ➢ Any action that compromises the security of information owned by an organization.
- ➢ Security Attacks are classified into two categories as in X.800 and RFC 2828.
- ➢ The Two categories of security attacks are :
    - o Passive Attacks
    - o Active Attacks

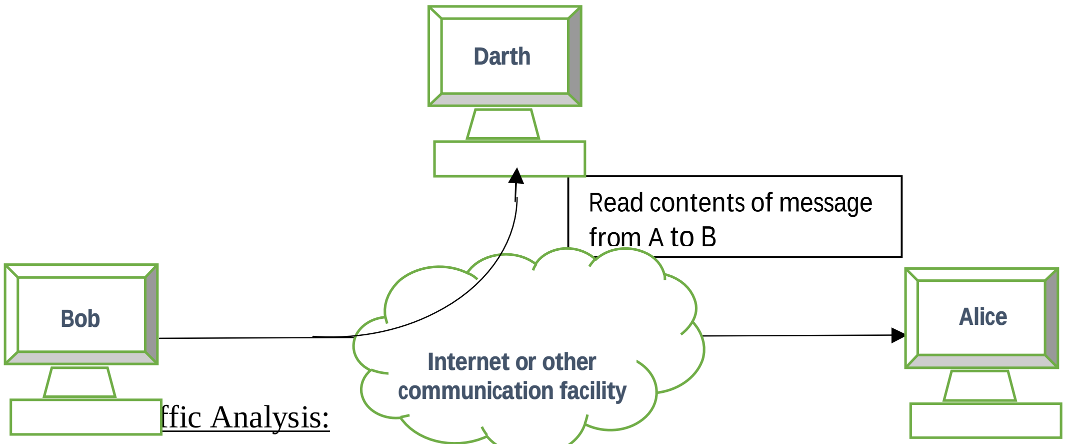
1.1.1 Passive Attacks

- ➢ A passive attack attempts to learn or make use of information from the system but does not alter system resources.
- ➢ Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions.

- The goal of the opponent is to obtain information that is being transmitted.
- Two types of passive attacks are the
  - Release of Message Contents
  - Traffic Analysis.

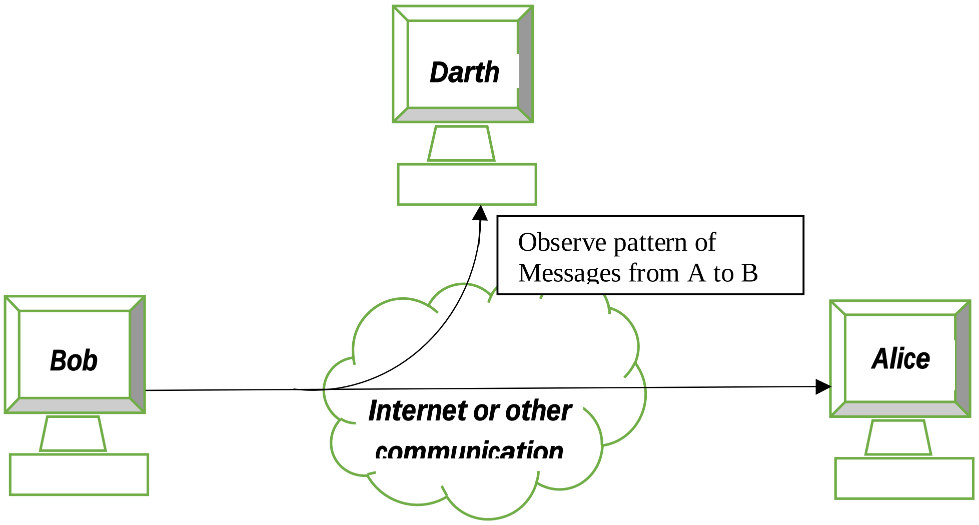
a. <u>Release of Message Contents:</u>

- ✓ A telephonic conversation, an E-mail message or a transferred file may contain confidential data.
- ✓ A passive attack may monitor the contents of these transmission.


Darth
Read contents of message from A to B
Bob
Internet or other communication facility
Alice

<u>ffic Analysis:</u>

- ✓ In this attack the eavesdropper analyzes the traffic, determines the location, identifies communicating hosts, and observes the frequency and length of message being exchanged.
- ✓ Using all these information they predict the nature of communication.
- ✓ All incoming and outgoing traffic of network is analyzed but not altered.


Darth
Observe pattern of Messages from A to B
Bob
Internet or other communication
Alice

- Passive attacks are very difficult to detect, because they do not involve any alteration of the data.

- Neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern.
- However, it is feasible to prevent the success of these attacks, usually by means of encryption.

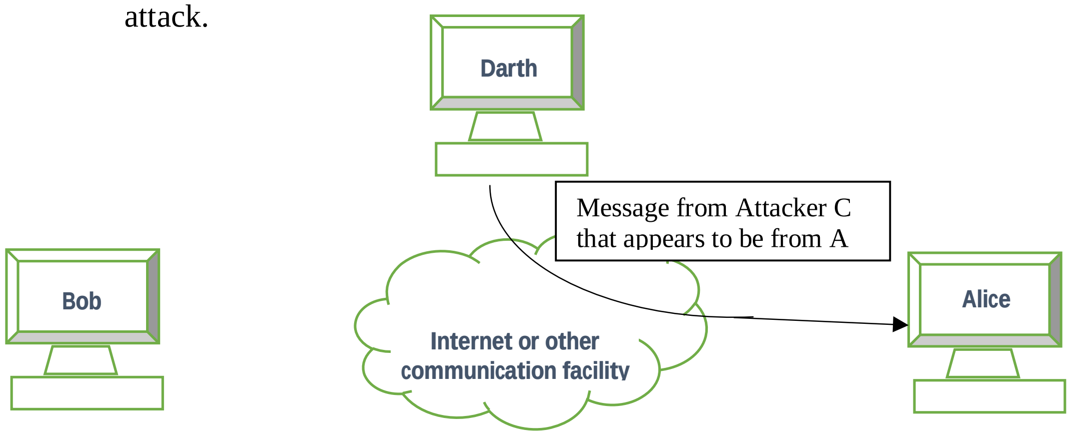
1.1.2 Active Attacks

Active attacks involve some alteration to resources of system or of the data stream or the creation of a false stream and caw be subdivided into four categories:

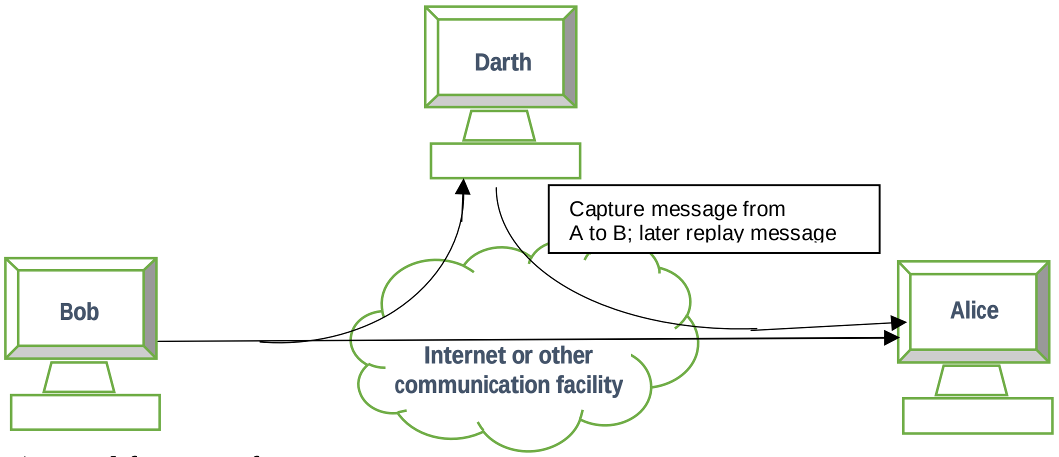
- Masquerade
- Replay
- Modification of Messages
- Denial of Service

a) Masquerade
   - It takes place when one entity pretends to be a different entity
   - A masquerade attack usually includes one of the other forms of active attack.

`

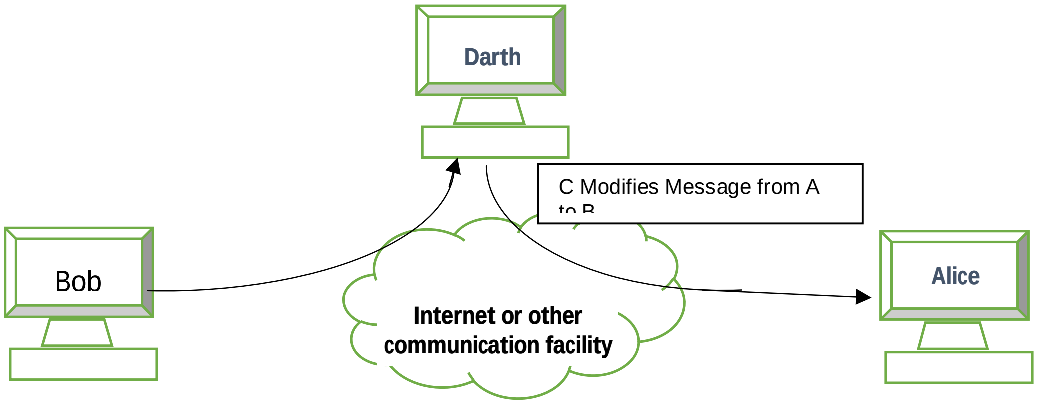

Darth
Message from Attacker C that appears to be from A
Bob
Internet or other communication facility
Alice

b) Replay
   - It involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect
   -



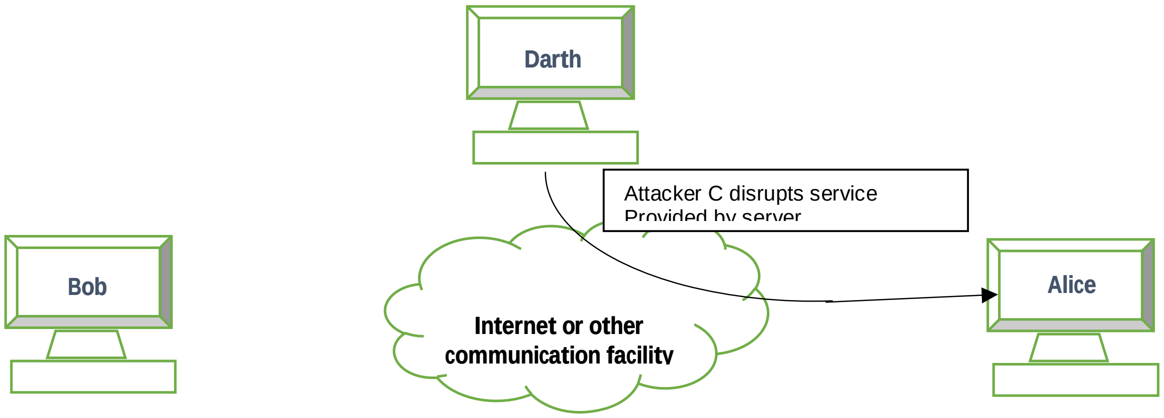
c) Modification of Messages

- simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect



d) Denial of Service

- prevents or inhibits the normal use or management of communications facilities



1.2 Security Service

- X.800 defines a security service as a service that is provided by a protocol layer of communicating open systems and that ensures adequate security of the systems or of data transfers

OR

- RFC 2828 defines it as: a processing or communication service that is provided by a system to give a specific kind of protection to system resources; security services implement security policies and are implemented by security mechanisms.
- X.800 divides these services into five categories and fourteen specific services
- Five categories in X.800 Services are
  - ✓ Authentication
  - ✓ Access Control
  - ✓ Data Confidentiality
  - ✓ Data Integrity
  - ✓ Nonrepudiation

1. Authentication: The assurance that the communicating entity is the one that it claims to be
   - o Peer Entity Authentication: Used in association with a logical connection to provide confidence in the identity of the entities connected.
   - o Data-Origin Authentication: In a connectionless transfer, provides assurance that the source of received data is as claimed.
2. Access Control: The prevention of unauthorized use of a resource

3. Data Confidentiality: The protection of data from unauthorized disclosure.
   - o Connection Confidentiality: The protection of all user data on a connection.
   - o Connectionless Confidentiality: The protection of all user data in a single data block
   - o Selective-Field Confidentiality: The confidentiality of selected fields within the user data on a connection or in a single data block.
   - o Traffic-Flow Confidentiality: The protection of the information that might be derived from observation of traffic flows
4. Data Integrity: The assurance that data received are exactly as sent by an authorized entity
   - o Connection Integrity with Recovery: Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.

- o <u>Connection Integrity without Recovery</u>: As above, but provides only detection without recovery.
- o <u>Selective-Field Connection Integrity</u>: Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.
- o <u>Connectionless Integrity</u>: Provides for the integrity of a single connectionless data block and may take the form of detection of data modification.
- o <u>Selective-Field Connectionless Integrity</u>: Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

5. Nonrepudiation: Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication
   1. control service and other security services.
   - o <u>Nonrepudiation</u>: Origin Proof that the message was sent by the specified party.
   - o <u>Nonrepudiation - Destination</u>: Proof that the message was received by the specified party.

6. Availability Service: Both X.800 and RFC 2828 define availability to be the property of a system and 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 available 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.
   - o 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

1.3 **Security Mechanism:** A process (or a device incorporating such a process) that is designed to detect, prevent, or recover from a security attack.

- ➢ Security mechanisms defined in X.800
  1. Specific Security Mechanisms
  2. Pervasive Security Mechanisms

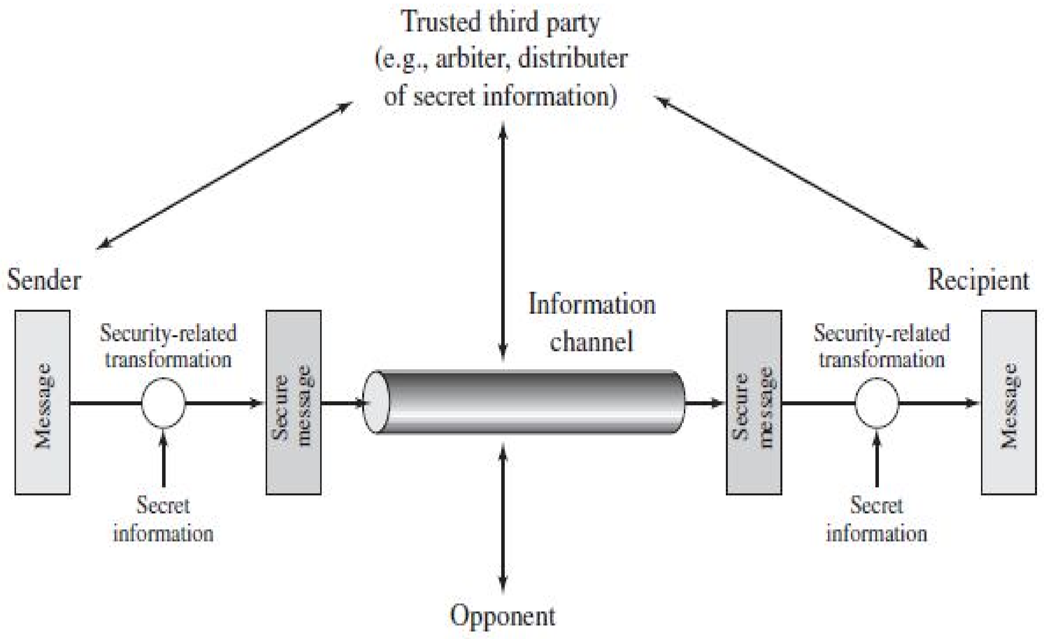
- Specific Security Mechanisms: May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services
  1. Encipherment
     - The use of mathematical algorithms to transform data into a form that is not readily intelligible.
     - The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys
  2. Digital Signature
     - Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).
  3. Access Control
     - A variety of mechanisms that enforce access rights to resources.
  4. Data Integrity
     - A variety of mechanisms used to assure the integrity of a data unit or stream of data units.
  5. Authentication Exchange
     - A mechanism intended to ensure the identity of an entity by means of information exchange
  6. Traffic Padding
     - The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.
  7. Routing Control
     - Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected
  8. Notarization
     - The use of a trusted third party to assure certain properties of a data exchange.
- Pervasive Security Mechanisms: Mechanisms that are not specific to any particular OSI security service or protocol layer
  1. Trusted Functionality
     - That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).
  2. Security Label
     - The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.
  3. Event Detection

- Detection of security-relevant events

4. Security Audit Trail
   - Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities
5. Security Recovery
   - Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

1.4 A model for Internetwork security

- A message is to be transferred from one party to another across some sort of Internet service.
- 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 by the two principals



- All the techniques for providing security have two components:
  - A security-related transformation on the information to be sent
  - Some secret information shared by the two principals
- A trusted third party may be needed to achieve secure transmission
- This general model shows that there are four basic tasks in designing a particular security service:
  - Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.

- Generate the secret information to be used with the algorithm.
- Develop methods for the distribution and sharing of the secret information.
- Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

1.5 Non Cryptographic protocol vulnerabilities:

In the early years of the internet was the way this protocol would be abused and impact it would have example of what such abuse can lead to is the Denial of service (DOS) attack.

Special features in the internet protocols such as TCP, UDP, and ICMP are exploited to launch DOS attacks

1.5.1 DOS:

Attack Types:

- The main purpose of Dos attack is to consume the resources of its victim to the point where it crawls to halt. An attacker may, for example, waste the computational power of its victim by inducting it to perform time-consuming cryptographic operations. Such operations are performed in setting up a security association using the IPSec protocol discussed in chapter 13. The attacker may also attempt to exhaust the memory of its victim or saturate its victim's access links to the internet. Dos attacks shot into prominence after several such attacks were launched on the websites of Amazon,Yahoo,eBay,etc...,in feburary 2000.

- We further highlight a number of Dos attack scenarios. Typically, a victim is flooded with packets that elicit some kind of response. Examples include the following:

- (1)An attacker sends thousands of TCP packets to its victim with the SYN flag set. The victim thinks that these are legitimate requests for TCP connection establishment (the first message of the three-way handshake).In response to each request, the victim reserves buffer space. Eventually, the victim's communication link and/or memory are exhausted. This is one of the most common Dos attacks and is referred to as a SYN flood.

- (2)An attacker sends a large number of UDP packets to non-listening ports on the victim. This causes the victim to respond with an ICMP "Host unreachable" message for each packets that its receives.

- (3)An attacker sends a very large number of ICMP"Echo Request" messages to the victims network. The destination IP address of these packets is the special broadcast address of the network, while the source IP address is the address of the victim. This causes the victim be inundated with: Echo Reply" messages form each host on its network. This is referred to as a *Smurf Attack.*

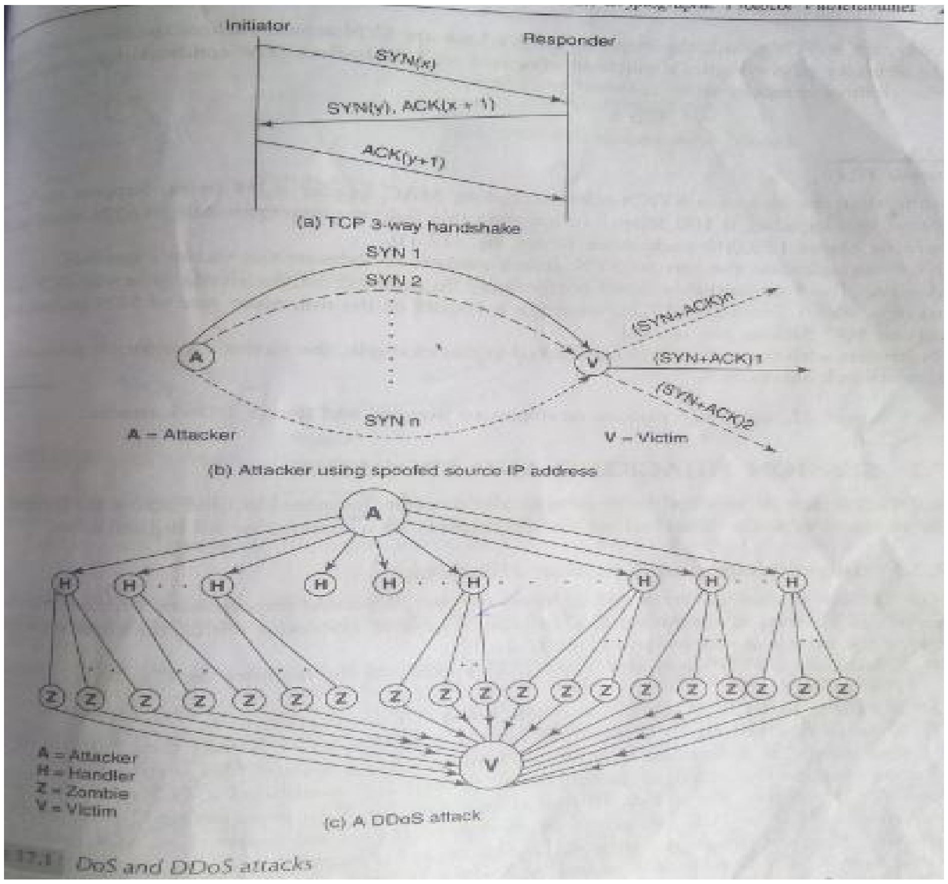
- ***Impact of SYN flooding***
- The TCP SYN flooding attack exploits a stateful nature of the three way hand shake of the TCP protocol, where in buffer space is reserved for each incoming connection request. The victim responds by sending a packet with a SYN and ACK flag set. However, the attacker typically uses a spoofed IP source address. So, the SYN + ACK message from the victim, is sent to a non existing or unsuspecting machine. In either case, the victim does not receive the third message – an ACK. The victim times out and resends the SYN +ACK. The timeout period and total number of retries are dependent on the operating system running on the victim's machine.

1.5.2 DDOS:

- To magnify the impact of the above attacks, the perpetrator may distribute the sources of the attack. A distributed DoS is also harder to detect compared to a DoS attack emanating from a single source. In a DDos attack, the brain behind the attack scans the internet to find the multiple vulnerable hosts called handlers and compromises them each handler .Each handler, in turn recruits many agents or zombies to launch the attack.
- Having multiple levels of attackers mean that more zombies can be co-opted thus amplifying the attack. For example, the controller may recruit 1000 handlers. If each handler controls 500 zombies, then we have a total of 500,000 zombies. The zombies are injected with the code that sends attack packets to the victim in a coordinated fashion to overwhelm it. In addition, the source IP addresses are spoofed to obscure the source of the attacks.
- A SYN flood is easy to launch and can have devastating consequences. We next quantity the impact of such an attack on the victims network bandwidth and on memory exhaustion.
- The following parameters are specific to the OS and communication link at the victim:
- l=communication bandwidth of the victims link.
- B=maximum number of buffers reserved for TCP connections

- T=maximum amount of time that a buffer can be reserved for a half-open TCP connection
- The following variables characterize the attack:
- r=aggregate rate at which the victim receives SYN attack packets from the attack sources.
- P=SYN attack packet size.
- The flood of TCP SYN attack packets will saturate the victim's link if the following inequality is satisfied

$$r * p \geq 1$$



17.1 DoS and DDoS attacks

1.5.3 Session hijacking and spoofing:

The DoS Attack is launched by geographically dispersed zombies located across the internet. Our next attack is typically launched within the confines of a computer or an organization.

I) <u>Impersonation and Session Hijacking</u>:

Kevin Mitnick devised an attack with in an attacker, X, Could impersonate a trusted client, C, to a server, S. The attack assumes that C, S, and X have IP addresses within the same network.

Step 1. X launches a SYN flood attack on C. This exhausts the memory on C's station allocated for TCP buffer.

Step 2. X then spoofs C's IP address and sets up a TCP connection to S.

Step 3. S thinks it is talking to C when, in fact, it is talking to X. X may then perform operations that only C is authorized.

- In Step2 X establishes a TCP connection with S. X spoofs C's IP address.
- In TCP Connection establishment, the second packet is an ACK packet from the responder, S, to the presumed initiator, C. This packet is received by C but is ignored because C is reeling from a SYN flood attack caused by X (step1) during which it ignores all incoming packets.
- To succeed, X will have to complete the three-way handshake that it initiated in step2. For this purpose it need to reed and increment the initial sequence number chosen by S. This number is denoted by y in the figure and include in the SYN+ACK response packet send by S to C. X can sniff this packet if it is on the same LAN as S.
- If X and S are in different LANs. In many early implementations of TCP, the initial sequence numbers were chosen very naively. For example, a station would increment y by a fixed amount for each new connection established by/to it. So, the initial sequence number chosen by S in the immediate past would be y-a, y-2a, y-3a... etc. In reverse chronological order.
- Once possibility is for X to repeatedly attempt to connect to C just to determine the algorithm used in choosing the initial sequence number. This could be done just before step2.
- The above attack succeeds because the server authenticates a client based on (1) the client's IP address and (2) the "ACK #" in the third message of the three-way handshake.

- The probability of success of this attack will be greatly reduced if initial sequence numbers are chosen randomly.
- If X is unable to sniff the second packet in the three-way handshake AND if the initial sequence number is truly random , it is hard for X to complete the three-way handshake so that X would not impersonate C.
- An attack similar to the above can be mounted to hijack a TCP connection. The difference between Mitnick's attack and TCP connection hijack is that, in the later two parties (say C and S) are already communicating. By flooding one of the parties (say C) and spoofing its address, an attacker may be able to continue the conversation with the other party, S. As before, the attacker makes S believe that it is talking to C.

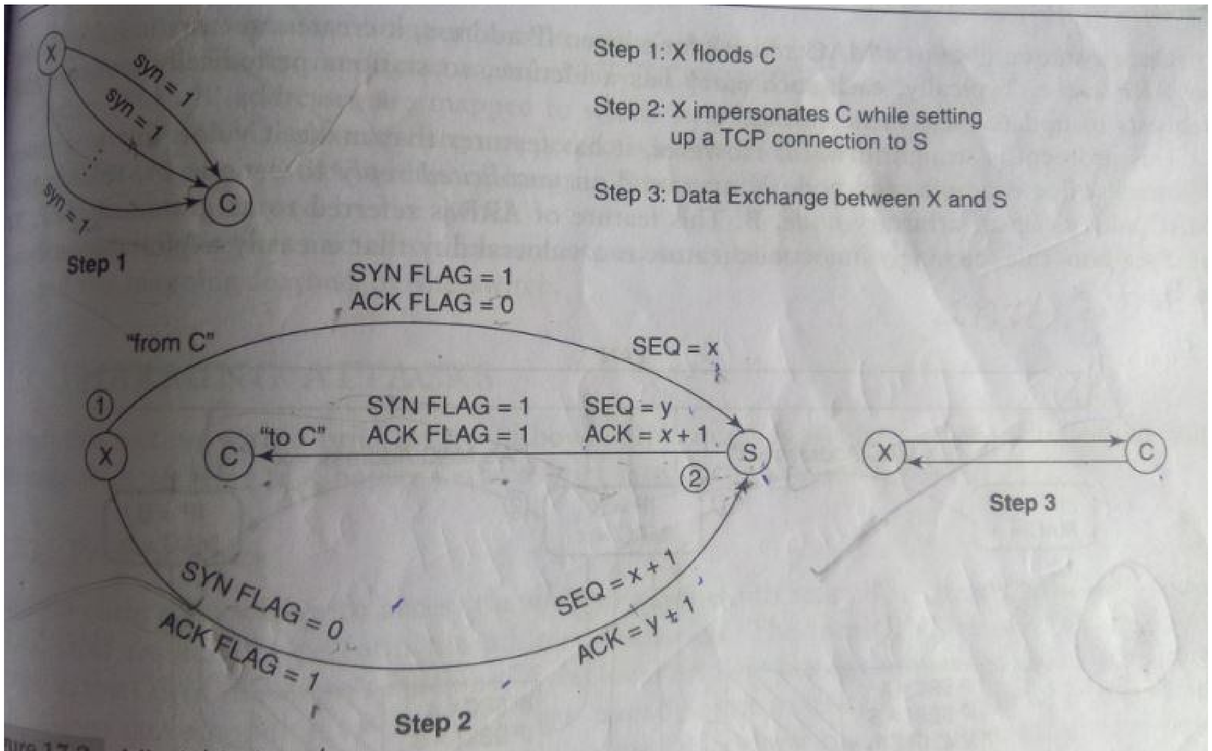


Figure 17.2 Mitnick's Attack

II) ARP Spoofing:

Address Resolution Protocol is used to resolve an IP address to a MAC address.

Consider two stations A and B on the same LAN. If A needs to send a packet to B, it not sufficient that A knows the IP address of B, A should also know B's MAC address.

For this purpose, A broadcasts an ARP query containing B's IP address. A station that has or knows B's MAC address responds directly to A.

Once a station obtains a MAC address for a given IP address, it creates an entry in its ARP table or ARP cache. Typically, each such cache has a lifetime, so stations periodically send out ARP requests to update their cache entries.

However, it has features that make it vulnerable to a verity of attacks. For example, any node X may send an unsolicited reply to node, A, regarding the MAC address of an arbitrary node, B. This feature of ARP is referred to as gratuitous ARP.

Consider an attacker, X, with IP address X, and MAC address, x.

It sends an unsolicited ARP response message to A containing the following: B’s MAC address is x.

X also sends an unsolicited ARP response message to B containing the following: A’s MAC address is x

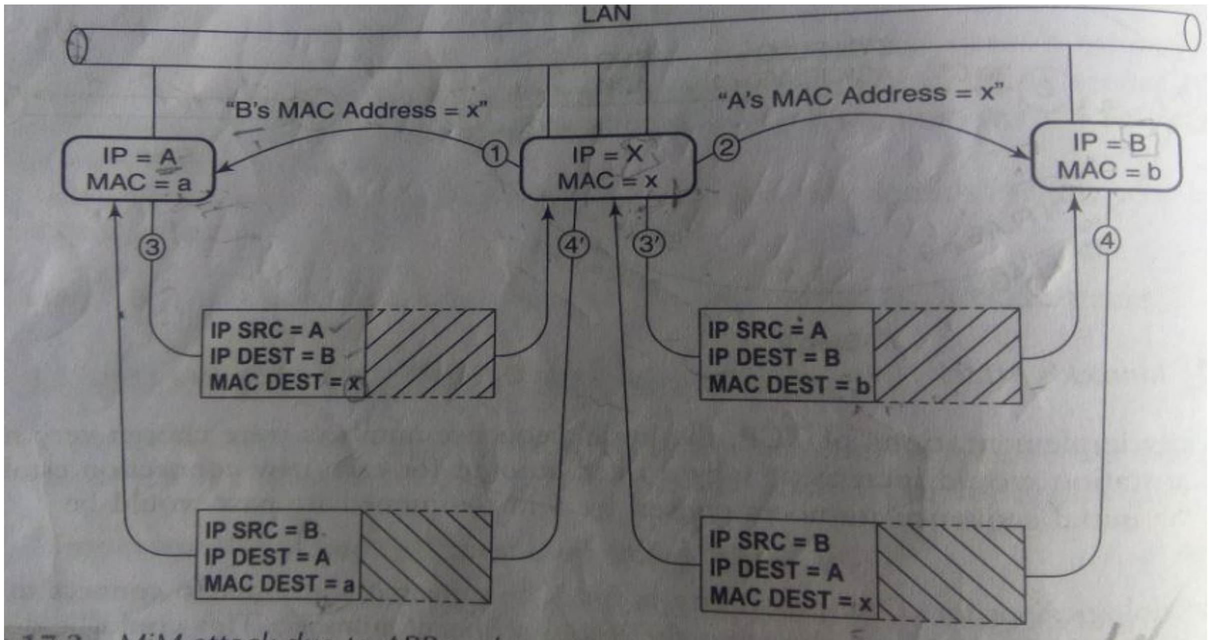
The unsolicited responses of X have created fake entries in the ARP cache of A and B. We say that their ARP caches have been poisoned.

Now, if A wishes to communicate with B, it will create a frame with destination MAC address =x. The LAN switch will forward this message to X. X may simply drop such frames thus choking off the connection from A to B. Alternatively, X may read and modify all such frames and then send them to B. Because B’s cache has also poisoned, X will be able to launch a man-in-the-middle attack.

Fortunately, there are solutions to the problem of ARP spoofing. One possibility is to have only authenticated ARP responses. This might require Kerberos style infrastructure or PKI.

A more radical solution is to use static ARP caches, which ignores send by random machines. Finally, intelligent switches may be designed that

- Learn which IP addresses are mapped to which switch port.
- Learn which MAC addresses are mapped to which switch port.
- Monitor IP address/MAC address pairing in Ethernet frames and check for inconsistency with what has been learned by the switch.
- Examine ARP replies and check for any inconsistency between the address in the ARP reply and the mapping learned by the switch.



17.3 MiM attack due to ARP cache poisoning

1.6 Software vulnerabilities

- In computer security, a vulnerability is a weakness which allows an attacker to reduce a system's information assurance.
- Vulnerability is the intersection of three elements: a system susceptibility or flaw, attacker access to the flaw, and attacker capability to exploit the flaw

1.6.1 Phishing

1 Phishing is the process of luring a victim to a fake website by clicking on a link. The victim usually encounters the link in an e-mail message sent to him or on a webpage being broused by him as in the following examples:

2 1. Click here www.luckeyDraw.com to claim your $1,000,000 prize!

3 2. urgent attention of all TrueBank Account Holders

4 Following a security breach, we wish to inform all our existing customers that we need to verify their account details. Kindly click here www.Truebank.com

5 once the victim clicks on the link such as that shown in the following figure , he/she may be induced to divulge sensitive information such as his credit card number or a password. For example, one of the highly published scams in recent time has been the phishing attack on-line banking.

6 Phishing attempts often use URLs that are very similar to the real URL. For example, the real URL may be www.TrueBank.com but the fake one may be www.TruBank.com. the fake URL corresponds to a website owned and operated by the attacker. once the user clicks on the fake link, he/she is presented with a webpage that has the same look and feel as that of the original website. He/she is then asked to enter login name and password. So as not to arouse any suspicion, he/she may be directed to the true site after entering his/her password. But, by then, attacker has harvested sensitive information like his/her password.

6.5.3 Buffer overflow

- ➢ A buffer overflow occurs when a program or process tries to store more data in a buffer (temporary data storage area) than it was intended to hold.
- ➢ Since buffers are created to contain a finite amount of data, the extra information - which has to go somewhere - can overflow into adjacent buffers, corrupting or overwriting the valid data held in them.
- ➢ In buffer overflow attacks, the extra data may contain codes designed to trigger specific actions, in effect sending new instructions to the attacked computer (for example, damage the user's files, change data, or disclose confidential information.)

1.6.2 Format String Attacks

C function such as Printf( ) take a format string as the first argument as in

printf("var1 in decimal is %d and var2 in hex is %x",var1,var2)

%d and %x are referred to as format specifiers. %d instructs printf to output var1 as a decimal number, while %x instructs printf to output var2 in hexadecimal format.

For each occurrence of %d and %x, printf( ) expects to see a variable name in its list of arguments.

Likewise for each occurrence of a %s or %n printf( ) expects to see pointer arguments. For example, printf(My name is %s",name); //her name is string variable, i.e., pointer to an array of characters

As it turns out, %s and %n are the format specifiers that can be used by an attacker to read and write arbitrary locations in memory. Here is an example of the use of the %n format specifier. printf("# of characters printed is %n",& count); printf("%d",count); the first printf( ) statement causes the number of bytes printed by that statement to be output to the variable, count. The second printf() statement then displays the number of bytes printed.

Instead of specifying the format string in the source program, it may be provided at runtime using printf(buf);

Here, a buffer, buf contains the format string. Of course, the format string could be just a simple string like "Hello" or it could contain one or more format specifiers. What would be printed by printf(buf) if buf was the string "Hello %d" instead?

To answer the above, recall that a calling program pushes a function's arguments on the stack before calling it. So, the program that calls

printf(" var1 in decimal is %d and var2 in hex is %x",var1,var2)

pushes the arguments of printf on the stack in the useal order –var2, then var1, and finally, a pointer to the format string. The printf function understands that the first format specifier in the format string corresponds to var1 and the second format specifier corresponds to var2.

Let us return to the earlier example printf(buf); where buf is "Hello %d". Here the calling program pushes only a single argument on the stack. However, printf(), on parsing the format string, encounters the %d. So it reaches in to the stack in an attempt to pick the second argument. This, however, is not a valid argument but may be a part of the callers stack frame.

For example with buf="Hello %d", printf(buf) may print the following Hello 93179

Function like printf() permit a variable number of arguments. The exact number is determined by parsing of format string which may only available at run time. A problem may arise if the number of arguments passed on the stack by the calling program is different from what is suggested by the format string. In particular, this could enable an attacker to read from an arbitrary location in memory.

1.6.3 SQL Injection

- SQL injection is a technique where malicious users can inject SQL commands into an SQL statement, via web page input.
- Injected SQL commands can alter SQL statement and compromise the security of a web application.
  or
- SQL injection is a type of web application security vulnerability in which an attacker is able to submit a database SQL command that is executed by a web application, exposing the back-end database.

- In order to run malicious SQL queries against a database server, an attacker must first find an input within the web application that is included inside of an SQL query.
- In order for an SQL Injection attack to take place, the vulnerable website needs to directly include user input within an SQL statement. An attacker can then insert a payload that will be included as part of the SQL query and run against the database server.
- The following server-side pseudo-code is used to authenticate users to the web application.

```
# Define POST variables
uname = request.POST['username']
passwd = request.POST['password']

# SQL query vulnerable to SQLi
sql = “SELECT id FROM users WHERE username=’” + uname + “’ AND password=’” + passwd + “’”

# Execute the SQL statement
database.execute(sql)
```

- The above script is a simple example of authenticating a user with a username and a password against a database with a table named users, and a username and password column.
- The above script is vulnerable to SQL Injection because an attacker could submit malicious input in such a way that would alter the SQL statement being executed by the database server.
- A simple example of an SQL Injection payload could be something as simple as setting the password field to password’ OR 1=1.
- This would result in the following SQL query being run against the database server.

```
SELECT id FROM users WHERE username=’username’ AND password=’password’ OR 1=1’
```

- An attacker can also comment out the rest of the SQL statement to control the execution of the SQL query further.

UNIT-I

Assignment-Cum-Tutorial Questions

SECTION-A

I. Objective Questions

1. Any action that compromises the security of information [ ]
a. Security Attack b. Security Mechanism c. Security Service
2. A process that is designed to detect, prevent, or recover from a security attack. [ ]
a. Security Mechanism b. Security Service
3. _________ attempts to learn or make use of information from the system but does not affect system resource [ ]
a. Passive Attack b. Active Attack
4. ______ attack attempts to alter system resources or affect their operation. [ ]
a. Passive Attack b. Active Attack
5. ______ takes place when one entity pretends to be a different entity
a. Masquerade b. Replay [ ]
b. c. Modification of Message d. Denial of Service
6. ______ involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect[ ]

a. Masquerade b. Replay
c. Modification of Message d. Denial of Service
7. _____ simply means that some portion of a legitimate message is altered [ ]
a. Masquerade b. Replay
c. Modification of Message d. Denial of Service
8. ______ prevents or inhibits the normal use or management of communications facilities [ ]
a. Masquerade b. Replay
c. Modification of Message d. Denial of Service
9. Message contents is released in _____ attack [ ]
a. release of message contents b. Denial of Service c. Traffic Analysis
10. _____ attack only traffic information is known [ ]
a. release of message contents b. Masquerade
c. Traffic Analysis d. Replay
11. The protection of data from unauthorized disclosure is _____ [ ]

a. Data Confidentiality b. Access Control c. Authentication

12. A denial of service attack: [ ]

a) can erase an entire Web site

b) does not have to occur over a network

c) is an intentional attempt to overload a web server or website

d) all of the above

13. Authentication is done for [ ]

a) Conventional encryption b) Scrambling data

c) Both a and b d) None of the above

14. Authentication is: [ ]

a) Verification of user's identification b) Verification of data

c) Both a and b d) None of the above

15. The process to discover plain text or key is known as: [ ]

a) Cryptanalysis b) Crypto design

c) Crypto processing d) Crypto graphic

16. Security mechanism is ensured in: [ ]

a) Detect attack b) Prevent attack

c)Recover from attack d) All of the above

17. In cryptography, what is cipher? [ ]

a) algorithm for performing encryption and decryption

b) encrypted message

c) both (a) and (b) d) none of the mentioned

18. _________ enhances the security of the data processing systems and the information transfers of an organization [ ]

a)Security Attack b) Security Mechanism c) Security Service

19. Cryptanalysis is used [ ]

a) to find some insecurity in a cryptographic scheme

b) to increase the speed

c) to encrypt the data d) none of the mentioned

20. When the firm's purpose for their information infrastructure is to make its data and information available to those who are authorized to use it, the firm is seeking the objective of: [ ]

a) confidentiality. b) availability. c) authorization. d)integrity.

SECTION-B

II. Descriptive Questions

1. What is meant by security attack? Explain different types of attacks with pictorial representation
2. Determine the security services required to counter various types of active and passive attacks.
3. Determine the security mechanisms required to provide various types of security services.
4. Draw the model of internetwork security and explain in detail.
5. Explain session hijacking and spoofing.
6. Explain phishing and buffer overflow attacks.
7. Write about format string attacks
8. Discuss about SQL injection techniques briefly.
9. Differentiate passive and active attacks.
10. Define threat and attack. What is the difference between both? List some examples of attacks which have arisen in real world cases.
11. Write a short note on DDoS.

INFORMATION SECURITY

UNIT II
Learning Material

Unit – II

Objectives:

- ➢ To introduce the symmetric cipher model
- ➢ To know different encryption algorithms and modes of operations.

Syllabus: Secret key Cryptography

Conventional encryption principles, conventional encryption algorithms, cipher block modes of operation, key distribution, approaches of message authentication, secure hash functions and HMAC.

Outcomes:

Students will be able to

- ➢ know symmetric encryption technique
- ➢ distinguish block and stream ciphers
- ➢ implement symmetric encryption algorithms

Basics of Cryptography

2.1 Symmetric cipher model

- Symmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the same key.
- It is also known as conventional encryption.
- Symmetric encryption transforms plaintext into cipher text using a secret key and an encryption algorithm.
- Using the same key and a decryption algorithm, the plaintext is recovered from the cipher text.
- A symmetric encryption scheme has five ingredients
    - Plaintext: This is the original intelligible message or data that is fed into the algorithm as input.
    - Encryption algorithm: The encryption algorithm performs various substitutions and transformations on the plaintext.
    - Secret key: The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm.
    - Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the secret key. The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.
    - • Decryption algorithm: This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

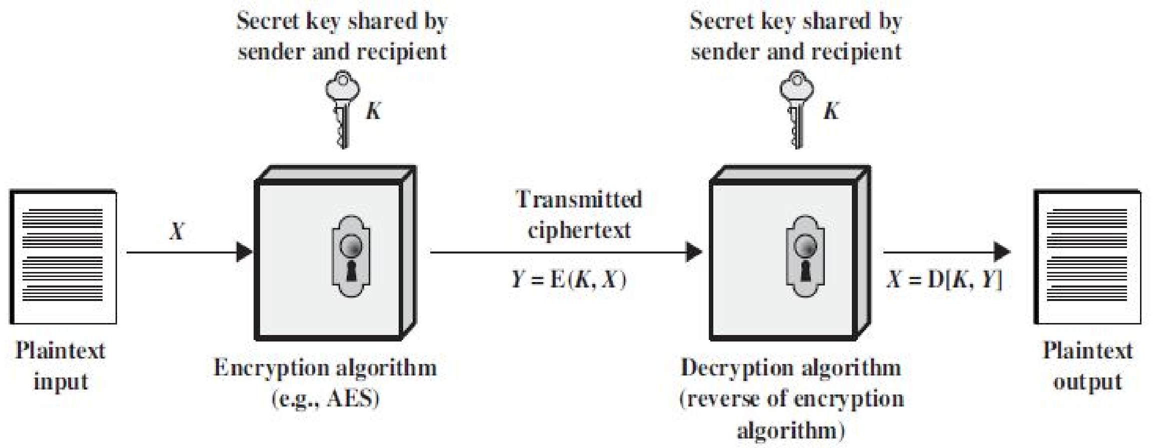
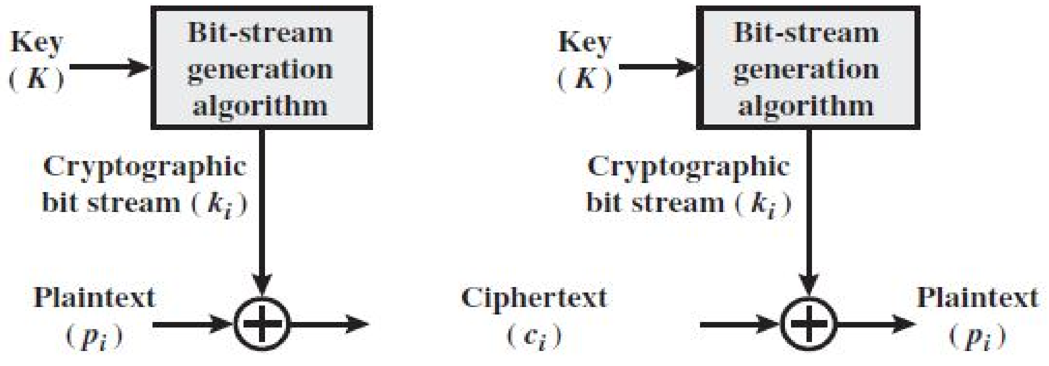

Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E(K, X)
X = D[K, Y]
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

Fig: Simplified Model of Symmetric Encryption

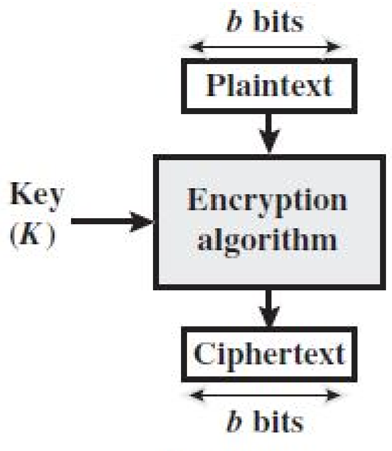
2.2 Block and Stream Ciphers

- **Stream cipher** is one that encrypts a digital data stream one bit or one byte at a time.
- Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher.
- If the cryptographic keystream is random, then this cipher is unbreakable by any means other than acquiring the keystream.
- However, the keystream must be provided to both users in advance via some independent and secure channel
- the bit-stream generator must be implemented as an algorithmic procedure, so that the cryptographic bit stream can be produced by both users.


Key
(K)
Bit-stream generation algorithm
Cryptographic bit stream (k_i)
Plaintext (p_i)
Key
(K)
Bit-stream generation algorithm
Cryptographic bit stream (k_i)
Ciphertext (c_i)
Plaintext (p_i)

(a) Stream cipher using algorithmic bit-stream generator

- A block cipher is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length.
- Many block ciphers have a Feistel structure.



(b) Block cipher

2.3 The Feistel Cipher

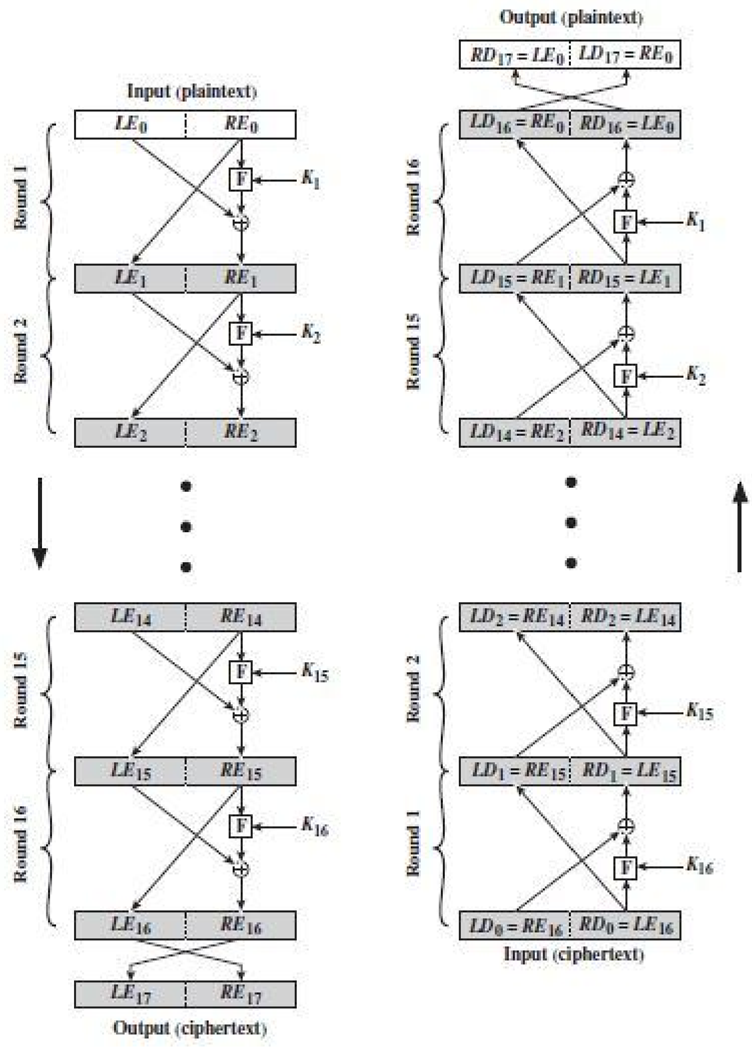
Feistel proposed the use of a cipher that alternates substitutions and permutations, where these terms are defined as follows:

- Substitution: Each plaintext element or group of elements is uniquely replaced by a corresponding ciphertext element or group of elements.
- Permutation: A sequence of plaintext elements is replaced by a permutation of that sequence. That is, no elements are added or deleted or replaced in the sequence, rather the order in which the elements appear in the sequence is changed.
- Feistel Cipher Structure
- Figure depicts the structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length 2w bits and a key K. The plaintext block is divided into two halves, L0 and R0.
- The two halves of the data pass through n rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs Li-1 and Ri-1, derived from the previous round, as well as a subkey Ki, derived from the overall K. In general, the subkeys Ki are different from K and from each other.
- All rounds have the same structure. A substitution is performed on the left half of the data. This is done by applying a round function F to the right half of the data and then taking the exclusive-OR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey Ki. Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data. This structure is a particular form of the substitution-permutation network (SPN) proposed by Shannon.
- The exact realization of a Feistel network depends on the choice of the following parameters and design features:

- Block size: Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion Traditionally, a block size of 64 bits has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.
- Key size: Larger key size means greater security but may decrease encryption/decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.
- Number of rounds: The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.
- Subkey generation algorithm: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
- Round function: Again, greater complexity generally means greater resistance to cryptanalysis. There are two other considerations in the design of a Feistel cipher
- Fast software encryption/decryption: In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern.
- Ease of analysis: Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength. DES, for example, does not have an easily analyzed functionality.
- Feistel Decryption Algorithm
- The process of decryption with a Feistel cipher is essentially the same as the encryption process. The rule is as follows: Use the ciphertext as input to the algorithm, but use the subkeys Ki in reverse order.
- That is, use Kn in the first round, Kn-1 in the second round, and so on until K1 is used in the last round. This is a nice feature because it means we need not implement two different algorithms, one for encryption and one for decryption.
- To see that the same algorithm with a reversed key order produces the correct result, consider Figure , which shows the encryption process going down the left-hand side and the decryption process going up the right-hand side for a 16-round algorithm (the result would be the same for any number of rounds). For clarity, we use the notation LEi and REi

for data traveling through the encryption algorithm and LDi and RDi for data traveling through the decryption algorithm. The diagram indicates that, at every round, the intermediate value of the decryption process is equal to the corresponding value of the encryption process with the two halves of the value swapped. To put this another way, let the output of the ith encryption round be LEi||REi (Li concatenated with Ri). Then the corresponding input to the (16 i) th decryption round is REi||LEi or, equivalently, RD16-i||LD16-i.

➢

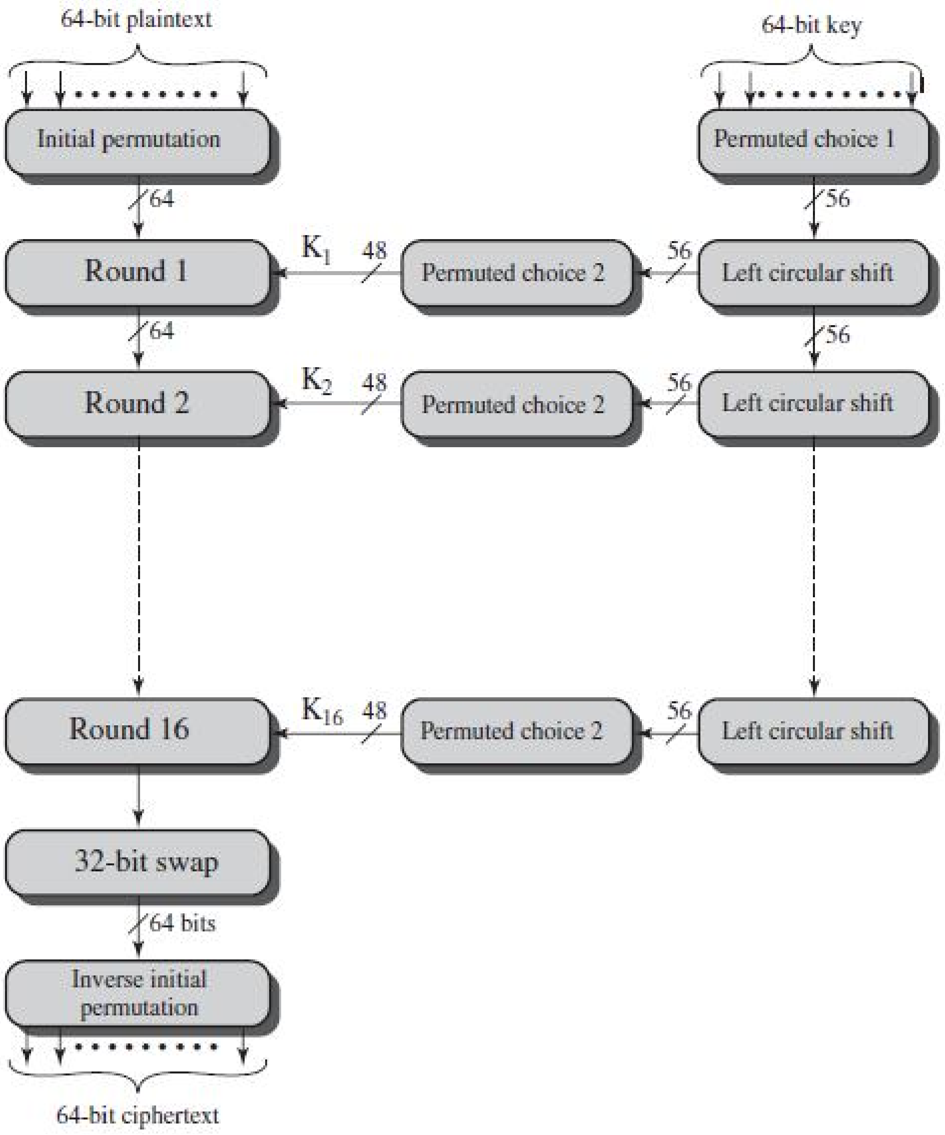


Feistel Encryption and Decryption(16 rounds)

2.4 Data Encryption Standard (DES)

➢ The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards.

- The algorithm is referred to as the Data Encryption Algorithm (DEA).
- Data are encrypted in 64-bit blocks using a 56-bit key.
- The algorithm transforms 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 plaintext must be 64 bits in length
  - the key is 56 bits in length

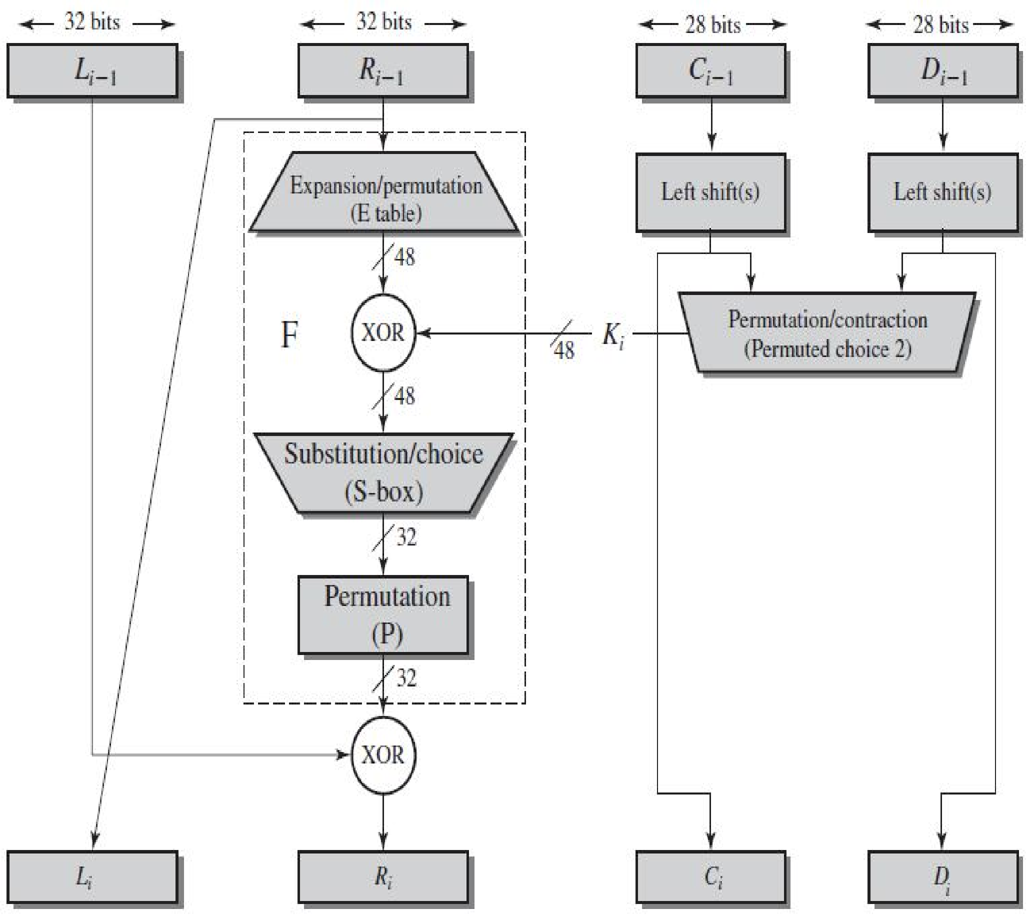


- Left-hand side of the figure has plaintext proceeded in three phases.
  - First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input

- It is followed by a phase consisting of sixteen rounds of the same function. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the preoutput.
- Finally, the preoutput is passed through a permutation that is the inverse of the initial permutation function, to produce the 64-bit ciphertext

➢ Right-hand portion of Figure 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 (K_i) 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.



- Details of Single Round
  - The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right).
  - the overall processing at each round can be summarized as:
  - $L_i = R_{i-1}$
  - $R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$
  - The round key K_i 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 resulting 48 bits are XORed with K_i.
  - This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted.
  - 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 as the first and last bits S_i of the input to box form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for S_i.
  - The middle four bits select one of the sixteen columns
  - The decimal value in the cell selected by the row and column is then converted to its 4-bit representation to produce the output.
  - 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.
- **DES Decryption**
  - Decryption uses the same algorithm as encryption, except that the application of the subkeys is reversed.
- The Avalanche Effect
  - A small change in either the plaintext or the key should produce a significant change in the cipher text.
  - 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 cipher text. This is referred to as the avalanche effect.

2.4.1 Strength of DES

The level of security provided by DES 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 2^{56} possible keys, which is approximately $7.2*10^{16}$ keys.
  - A brute-force attack is impractical on DES.
  - On average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher.
- The Nature of the DES Algorithm
  - The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration.
  - A number of regularities and unexpected behaviors of the S-boxes have been discovered. Despite this, no one has so far succeeded in discovering the supposed fatal weaknesses in the S-boxes.
- Timing Attacks
  - A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.
  - DES appears to be fairly resistant to a successful timing attack.

2.5 Block Cipher Design Principles

The three critical aspects of block cipher design:

- The number of rounds,
- Design of the function F,
- Key scheduling.

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

    - The heart of a DES is the function F which relies on the use of
      S-boxes.
    - The function F provides the element of confusion.
    - Thus, it must be difficult to "unscramble" the substitution performed by F.
    - The criterion is that F must be nonlinear. The more nonlinear F, the more difficult any type of cryptanalysis will be.
  - S-Box Design
    - A change to the input vector to an S-box should result in random-looking changes to the output.
    - The relationship should be nonlinear and difficult to approximate with linear functions
    - Random: Use some pseudorandom number generation or some table of random digits to generate the entries in the S-boxes.
    - Random with testing: Choose S-box entries randomly, then test the results against various criteria, and throw away those that do not pass.
    - Human-made: This is a more or less manual approach with only simple mathematics to support it.
    - Math-made: Generate S-boxes according to mathematical principles. By using mathematical construction, S-boxes can be constructed that offer proven security against linear and differential cryptanalysis, together with good diffusion
- Key scheduling.
  - The key is used to generate one subkey for each round.
  - In general, we would like to select subkeys to maximize the difficulty of deducing individual subkeys and the difficulty of working back to the main key.
  - The key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion

DES Design Criteria: focus is on the design of the S-boxes and on the P function that takes the output of the S-boxes.

- The criteria for the S-boxes are as follows

1. No output bit of any S-box should be too close a linear function of the input bits.
2. Each row of an S-box should include all 16 possible output bit combinations.
3. If two inputs to an S-box differ in exactly one bit, the outputs must differ in at least two bits
4. If two inputs to an S-box differ in the two middle bits exactly, the outputs must differ in at least two bits.
5. If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must not be the same.
6. For any nonzero 6-bit difference between inputs, no more than eight of the 32 pairs of inputs exhibiting that difference may result in the same output difference.
7. This is a criterion similar to the previous one, but for the case of three S-boxes.

- The criteria for the permutation P are as follows.
  1. The four output bits from each S-box at round are distributed so that two of them affect "middle bits" of round $(i + 1)$ and the other two affect end bits.
  2. The four output bits from each S-box affect six different S-boxes on the next round, and no two affect the same S-box.
  3. For two S-boxes j, k, if an output bit from S_j affects a middle bit of S_k on the next round, then an output bit from S_k cannot affect a middle bit of S_j. This implies that, for j=k, an output bit from S_j must not affect a middle bit of S_j.

2.5.1 Blowfish

- a symmetric block cipher designed by Bruce Schneier in 1993/94
- characteristics
  - fast implementation on 32-bit CPUs
  - compact in use of memory
  - simple structure eases analysis/implemention
  - variable security by varying key size
- has been implemented in various products

Blowfish Key Schedule

- uses a 32 to 448 bit key
- used to generate
  - 18 32-bit subkeys stored in K-array K_j
  - four 8x32 S-boxes stored in $S_{i,j}$
- key schedule consists of:

- initialize P-array and then 4 S-boxes using pi
- XOR P-array with key bits (reuse as needed)
- loop repeatedly encrypting data using current P & S and replace successive pairs of P then S values
- requires 521 encryptions, hence slow in rekeying

Blowfish Encryption

- uses two primitives: addition & XOR
- data is divided into two 32-bit halves L_0 & R_0

```
for $i$ = 1 to 16 do
$R_i = L_{i-1}$ XOR $P_i$;
$L_i = F[R_i]$ XOR $R_{i-1}$;
$L_{17} = R_{16}$ XOR $P_{18}$;
$R_{17} = L_{16}$ XOR $i_{17}$;
```

- where

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

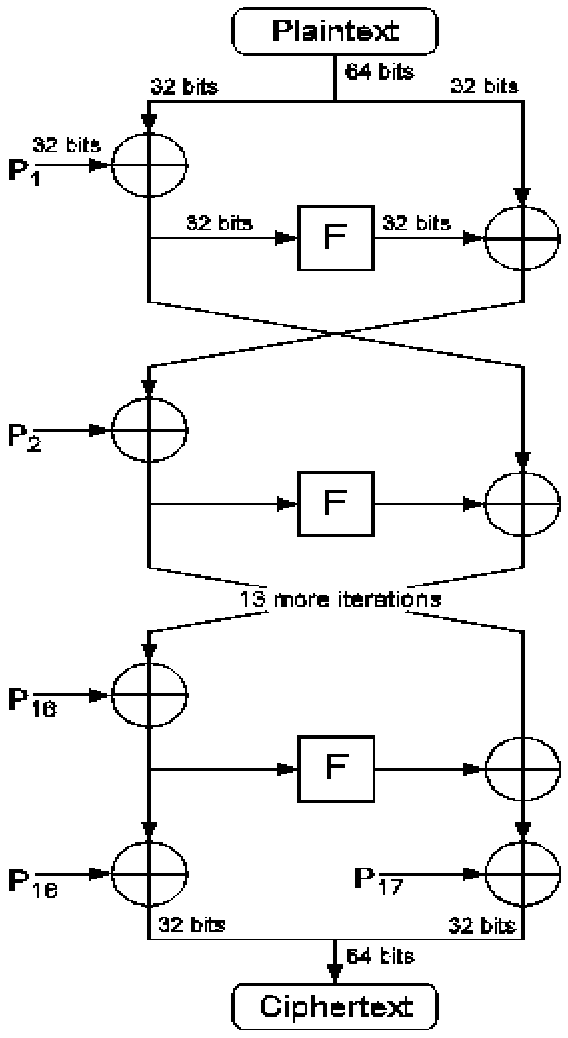


Figure: Blowfish Encryption

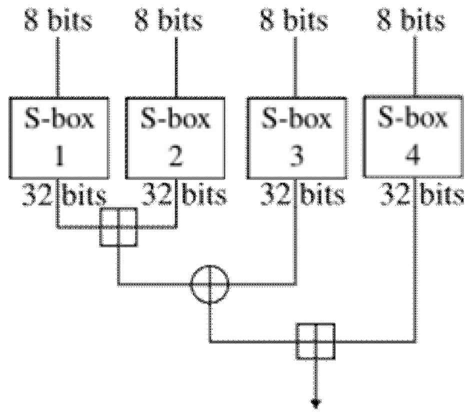


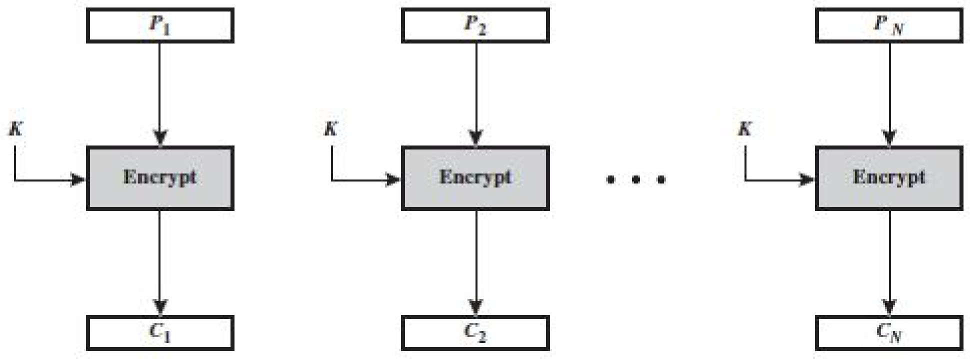
Figure: Round function of Blowfish

2.6 Modes of operation

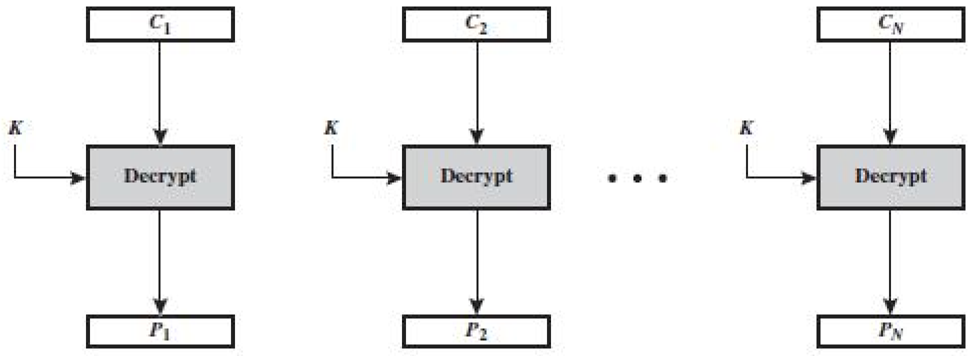
- A block cipher takes a fixed-length block of text of length bits and a key as input and produces a -bit block of ciphertext.
- If the amount of plaintext to be encrypted is greater than b bits, then the block cipher can still be used by breaking the plaintext up into -bit blocks
- To apply a block cipher in a variety of applications, five modes of operation have been defined by NIST
- In essence, a mode of operation is a technique for enhancing the effect of a cryptographic algorithm or adapting the algorithm for an application, such as applying a block cipher to a sequence of data blocks or a data stream.
- The Block Cipher modes of operations are:
    - Electronic Codebook (ECB)
    - Cipher Block Chaining (CBC)
    - Cipher Feedback (CFB)
    - Output Feedback (OFB)
    - Counter (CTR)

2.6.1 Electronic Codebook (ECB)

- Plaintext is handled one block at a time and each block of plaintext is encrypted using the same key.
- The term codebook is used because, for a given key, there is a unique ciphertext for every -bit block of plaintext.
- Message is broken into independent blocks of b-bit size which are encrypted
- Decryption is performed one block at a time, always using the same key.



(a) Encryption

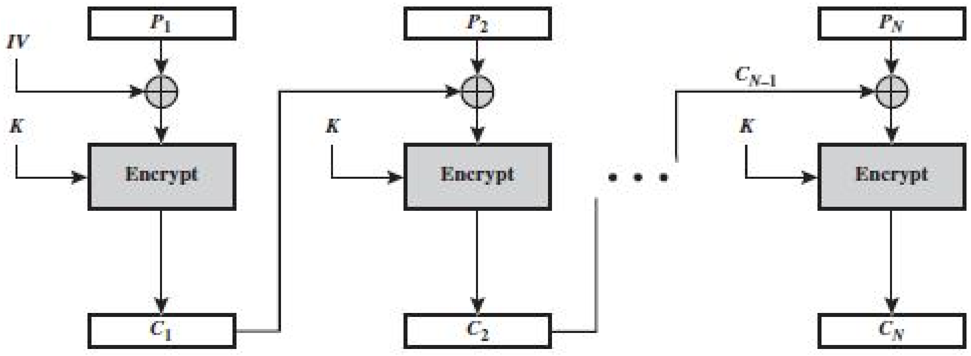


(b) Decryption

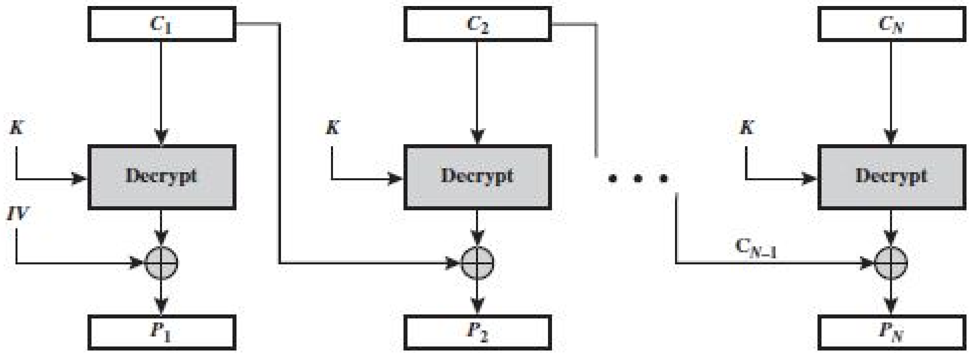
- The plaintext consists of a sequence of b-bit blocks, P_1, P_2.... P_N.
- The corresponding sequence of ciphertext blocks is C_1, C_2 ..... C_N
- ECB mode is defined as:
  - $C_j = E(K, P_j)$ $j = 1, \ldots, N$ (Encryption)
  - $P_j = D(K, C_j)$ $j = 1, \ldots, N$ (Decryption)
- The ECB method is ideal for a short amount of data, such as an encryption key
- For lengthy messages, the ECB mode may not be secure
- weakness due to encrypted message blocks being independent

2.6.2 Cipher Block Chaining (CBC)

- the input to the encryption algorithm is the XOR of the current plaintext block and the preceding ciphertext block.
- the same key is used for each block
- To produce the first block of ciphertext, an initialization vector (IV) is XORed with the first block of plaintext



(a) Encryption

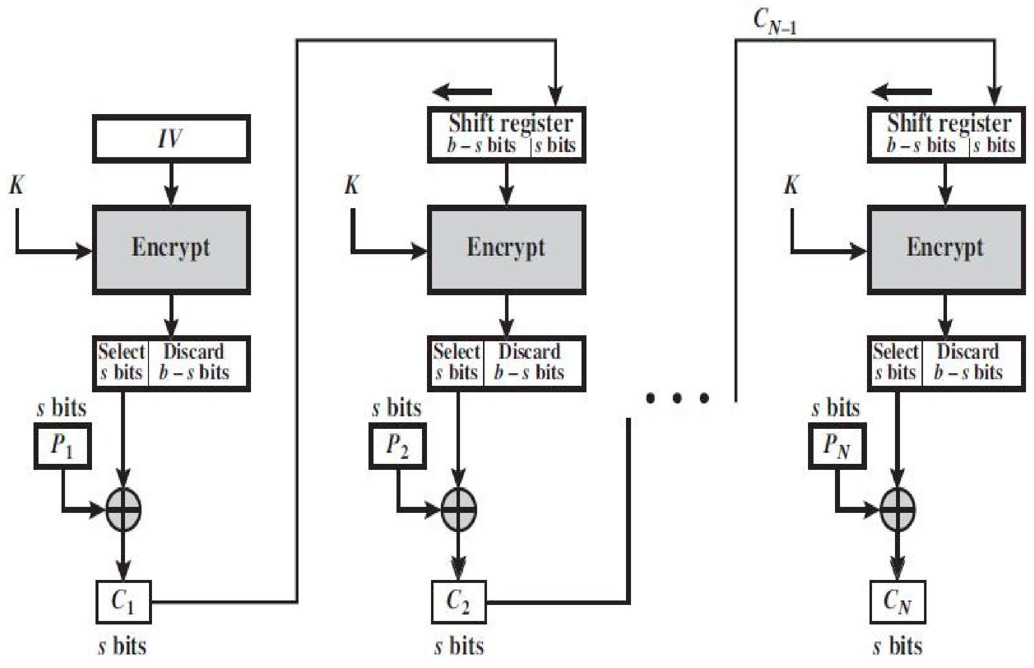


(b) Decryption

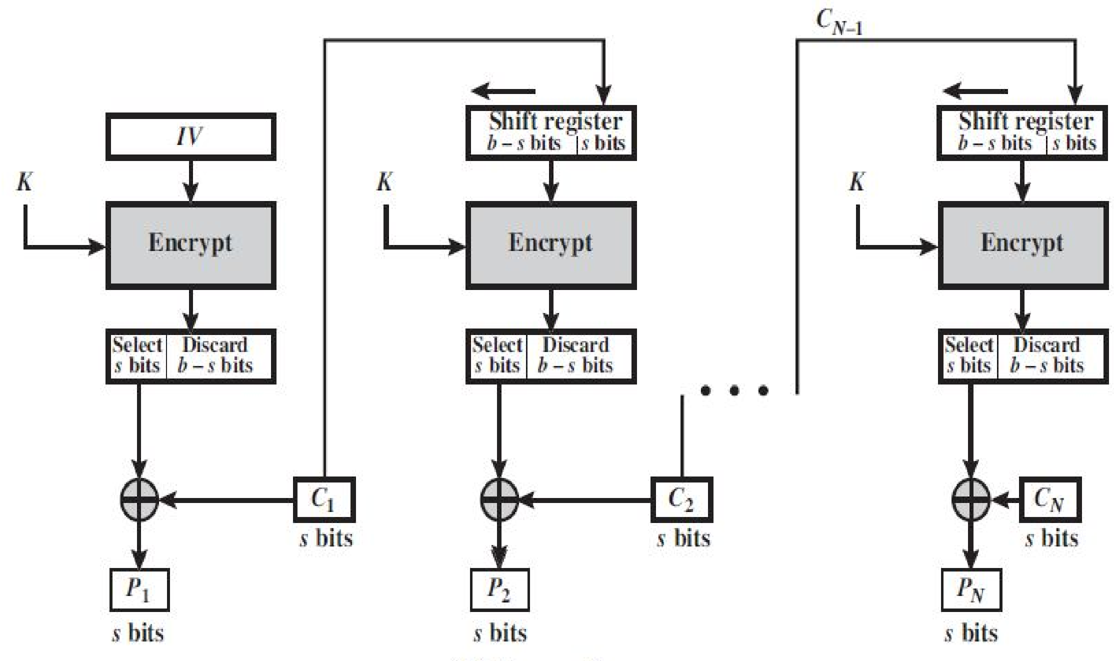
- On decryption, the IV is XORed with the output of the decryption algorithm to recover the first block of plaintext.
- The IV is a data block that is that same size as the cipher block.
- The IV must be known to both the sender and receiver but be unpredictable by a third party.
- We can define CBC mode as
    - Encryption
        - $C_1 = E(K, [P_1 \oplus IV])$
        - $C_j = E(K, [P_j \oplus C_{j-1}])\ j = 2, \ldots N$
    - Decryption
        - $P_1 = D(K, C_1) \oplus IV$
        - $P_j = D(K, C_j) \oplus C_{j-1}\ j = 2, \ldots N$
- each ciphertext block depends on all message blocks thus a change in the message affects all ciphertext blocks after the change as well as the original block

2.6.3 Cipher Feedback (CFB)

- the plaintext is divided into segments of s bits
- The input to the encryption function is a b-bit shift register that is initially set to some initialization vector (IV).
- The leftmost s bits of the output of the encryption function are XORed with the first segment of plaintext P_1 to produce the first unit of ciphertext C_1.
- The contents of the shift register are shifted left by s bits, and C_1 is placed in the rightmost s bits of the shift register
- Process continues until all plaintext units have been encrypted.
- Encryption $C_1 = P_1 \oplus \text{MSBs}[E(K, IV)]$
- Decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext.
- Decryption $P_1 = C_1 \oplus \text{MSB}s[E(K, IV)]$



(a) Encryption

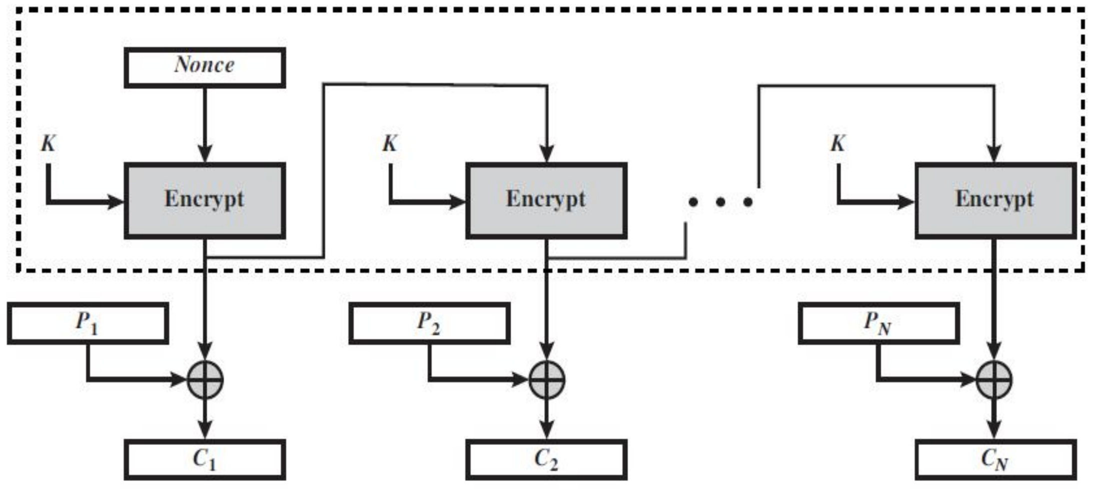


(b) Decryption

- This is most common stream mode
- The disadvantage is errors propagate for several blocks after the error occurred.

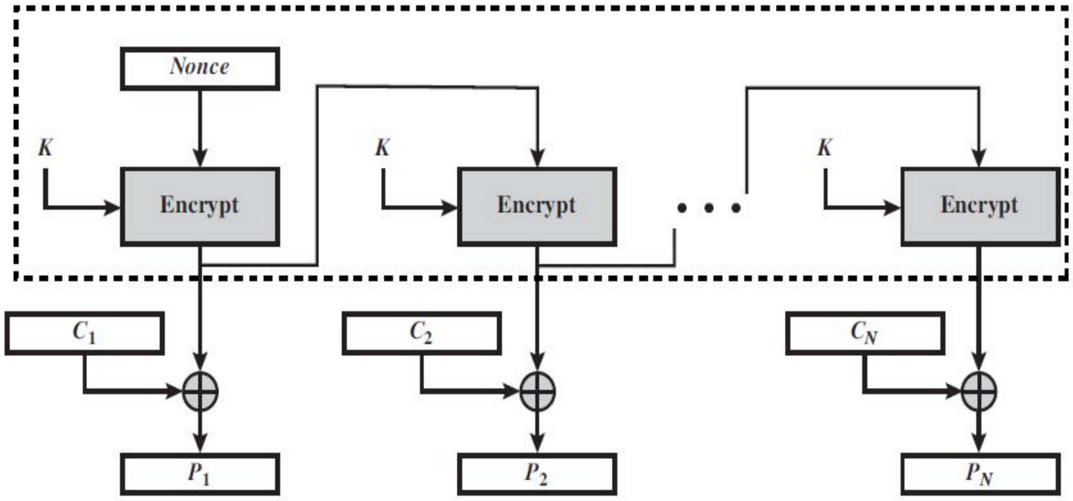
2.6.4 Output Feedback (OFB)

- The output of the encryption function that is fed back to the shift register in OFB.
- message is treated as a stream of bits
- OFB mode operates on full blocks of plaintext and ciphertext



(a) Encryption

- Encryption can be expressed as $C_j = P_j \oplus E(K, [C_{j-l} \oplus P_{j-1}])$



(b) Decryption

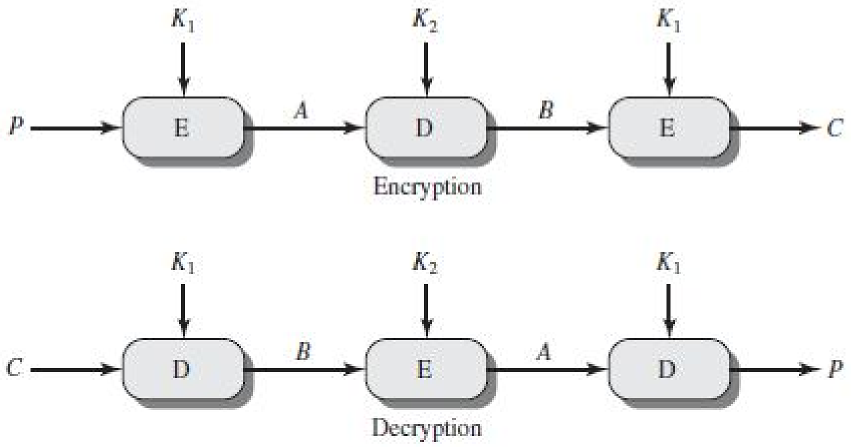
- Decryption can be expressed as $P_j = C_j \oplus E(K, [C_{j-1} \oplus P_{j-1}])$

2.7 Triple DES (3DES)

Triple DES makes use of three stages of the DES algorithm, using a total of two or three distinct keys:

- Triple DES with Two Keys
- Triple DES with Three Keys

- Triple DES with Two Keys
  - A triple encryption method that uses only two keys
  - The function follows an encrypt-decrypt-encrypt (EDE) sequence as below



  - Encryption is : $C = E(K_1, D(K_2, E(K_1, P)))$
  - Decryption is : $P = D(K_1, E(K_2, D(K_1, C)))$

  - There is no cryptographic significance to the use of decryption for the second stage.
  - Its only advantage is that it allows users of 3DES to decrypt data encrypted by users of the older single DES
  - 3DES with two keys is a relatively popular alternative to DES and has been adopted for use in the key management standards.
- Triple DES with Three Keys
  - Three-key 3DES has an effective key length of 168 bits and is defined as : $C = E(K_3, D(K_2, E(K_1, P)))$
  - Backward compatibility with DES is provided by putting $K_3 = K_2$ or $K_1 = K_2$
  - A number of Internet-based applications have adopted three-key 3DES, including PGP and S/MIME.

2.7 Key Distribution

For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key. Therefore, the strength of any cryptographic system rests with the key distribution technique, a term that refers to the means of delivering a
key to two parties that wish to exchange data, without allowing others to see the key.
Key distribution can be achieved in a number of ways. For two parties A and B, There are the following options:
1. A key could be selected by A and physically delivered to B.
2. A third party could select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party could transmit the new key to the other, using the old key to encrypt the new key.
4. If A and B each have an encrypted connection to a third party C, C could deliver a key on the encrypted links to A and B.

Options 1 and 2 call for manual delivery of a key. For link encryption, this is a reasonable requirement, because each link encryption device is only going to be exchanging data with its partner on the other end of the link. However, for end-to end encryption over a network, manual delivery is awkward.

Option 3 is a possibility for either link encryption or end-to-end encryption, but if an attacker ever succeeds in gaining access to one key, then all subsequent keys are revealed. Even if frequent changes are made to the link encryption keys, these should be done manually.

For **option 4**, two kinds of keys are used:

- **Session key:**When two end systems (hosts, terminals, etc.) wish to communicate, they establish a logical connection (e.g., virtual circuit). For the duration of that logical connection, called a session, all user data are encrypted with a one-time session key. At the conclusion of the session the session key is destroyed.
- **Permanent key:** A permanent key is a key used between entities for the purpose of distributing session keys.

A necessary element of option 4 is a **key distribution center (KDC)**. The KDC determines which systems are allowed to communicate with each other.When permission is granted for two systems to establish a connection, the key distribution center provides a one-time session key for that connection.

In general terms, the operation of a KDC proceeds as follows:

1. When host A wishes to set up a connection to host B, it transmits a connectionrequest packet to the KDC. The communication between A and the KDC is encrypted using a master key shared only by A and the KDC.
2. If the KDC approves the connection request, it generates a unique one-time session key. It encrypts the session key using the permanent key it shares with A and delivers the encrypted session key to A. Similarly, it encrypts the session key using the permanent key it shares with B and delivers the encrypted session key to B.
3. A and B can now set up a logical connection and exchange messages and data,all encrypted using the temporary session key.

2.8 Approaches to message authentication

Encryption protects against passive attack (eavesdropping). A different requirement is to protect against active attack (falsification of data and transactions). Protection against such attacks is known as message authentication.

2.8.1 Authentication Using Conventional Encryption

It would seem possible to perform authentication simply by the use of symmetric encryption. If we assume that only the sender and receiver share a key (which is as it should be), then only the genuine sender would be able to encrypt a message successfully for the other participant, provided the receiver can recognize a valid message. Furthermore, if the message includes an error-detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper. If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit.

2.8.2 Message Authentication without Message Encryption

The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.

Three situations in which message authentication without confidentiality is preferable:

1. There are a number of applications in which the same message is broadcast to a number of destinations.Two examples are notification to users that the network is now unavailable and an alarm signal in a control center. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity.Thus, the message must be broadcast in plaintext with an associated message authentication tag. The responsible system 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 with messages being chosen at random for checking.
3. Authentication of a computer program in plaintext 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 tag were attached to the program, it could be checked whenever assurance is required of the integrity of the program.\

2.9 Secure Hash Function

The one-way hash function, or secure hash function, is important not only in message authentication but in digital signatures.

2.9.1 The SHA Secure Hash Function

the most widely used hash function has been the Secure Hash Algorithm (SHA). Indeed, because virtually every other widely used hash function had been found to have substantial cryptanalytic weaknesses, SHA was more or less the last remaining standardized hash algorithm by 2005. SHA was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993.

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

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

Step 2 Append length: A block of 128 bits is appended to the message.This block is
treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding). The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. In Figure 3.4, the expanded message is represented as the sequence of 1024-bit blocks *M*1, *M*2, . . ., *MN*, so that the total length of the expanded message is *N* × 1024 bits.

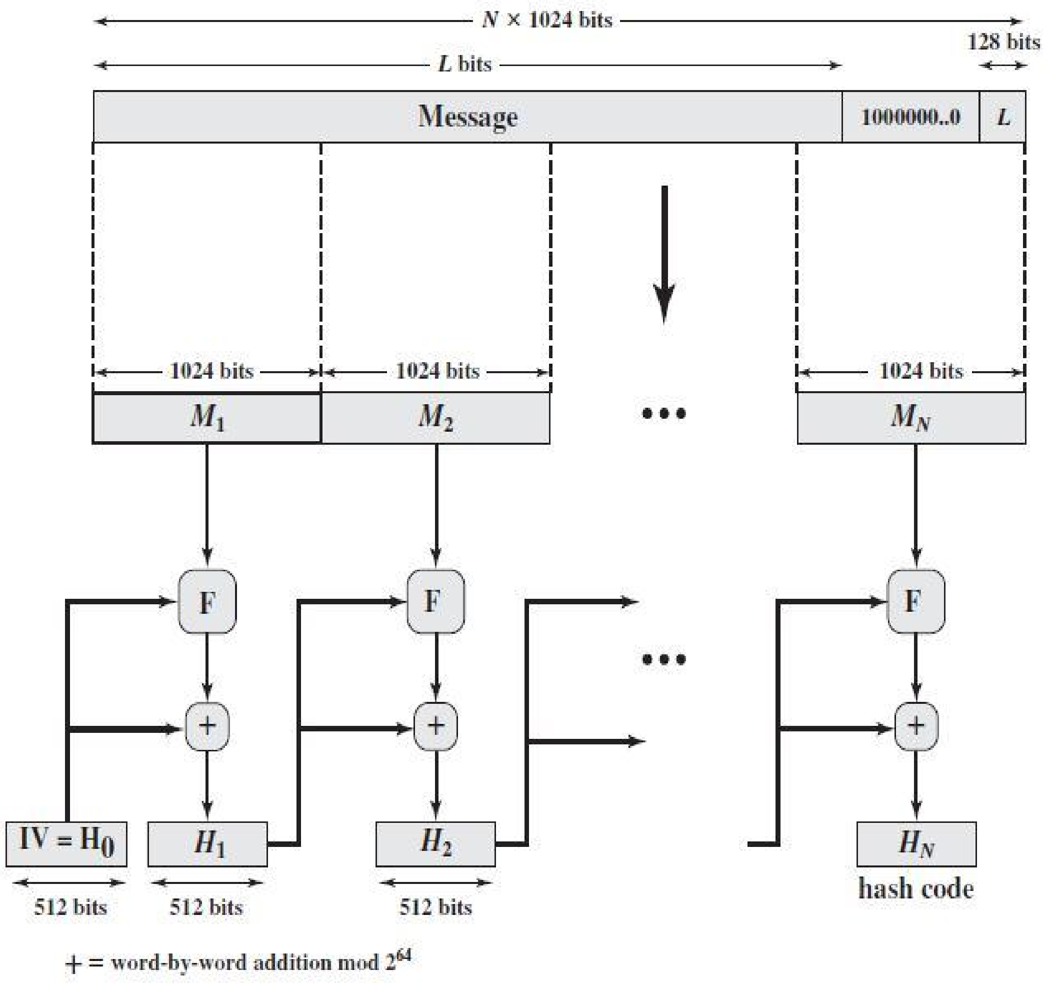
Step 3 Initialize hash buffer: A 512-bit buffer is used to hold intermediate and final results of the hash function.The buffer can be represented as eight 64-bit registers (*a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*).These registers are initialized to the following 64-bit integers (hexadecimal values):

a 6A09E667F3BCC908 *e* 510E527FADE682D1
b BB67AE8584CAA73B *f* 9B05688C2B3E6C1F
c 3C6EF372FE94F82B *g* 1F83D9ABFB41BD6B
d A54FF53A5F1D36F1 *h* 5BE0CD19137E2179

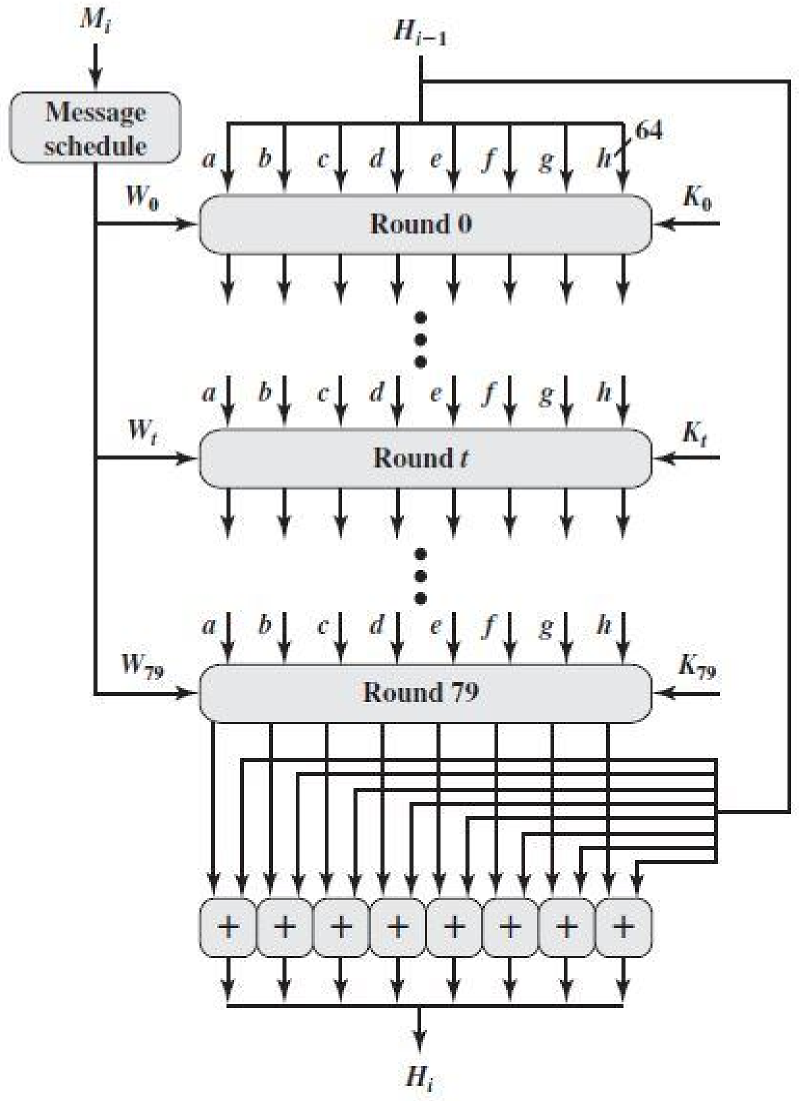
These values are stored in big-endian format, which is the most significant byte of a word in the low-address (leftmost) byte position.These words were obtained by taking the first sixty-four bits of the fractional parts of the square roots of the first eight prime numbers.

Step 4 Process message in 1024-bit (128-word) blocks: The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F in Figure 3.4.The logic is illustrated in Figure 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*). Each round also makes use of an additive constant *Kt*, where 0 … *t* … 79 indicates one of the 80 rounds.These words represent the first 64 bits of the fractional parts of the cube roots of the first 80 prime numbers. The constants provide a “randomized” set of 64-bit patterns, which should eliminate any regularities in the input data. The output of the 80th round is added to the input to the first round (*Hi*-1) to produce *Hi.*The addition is done independently for each of the eight words in the buffer with each of the corresponding words in *Hi* 1, using addition modulo 264.

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



Message Digest Generation Using SHA-512



SH
A-512 Processing of a Single 1024-Bit Block

The SHA-512 algorithm has the property that every bit of the hash code is a function of every bit of the input. The complex repetition of the basic function F produces results that are well mixed; that is, it is unlikely that two messages chosen at random, even if they exhibit similar regularities, will have the same hash code. Unless there is some hidden weakness in SHA-512, which has not so far been published, the difficulty of coming up with two messages having the same message digest is on the order of 2^256 operations, while the difficulty of finding a message with a given digest is on the order of 2^512 operations.

2.10 HMAC

A hash function such as SHA-1 was not designed for use as a MAC and cannot be used directly for that purpose because it does not rely on a secret key. There have been a number of proposals for the incorporation of a secret key into an existing hash algorithm. The approach that has received the most support is HMAC.

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

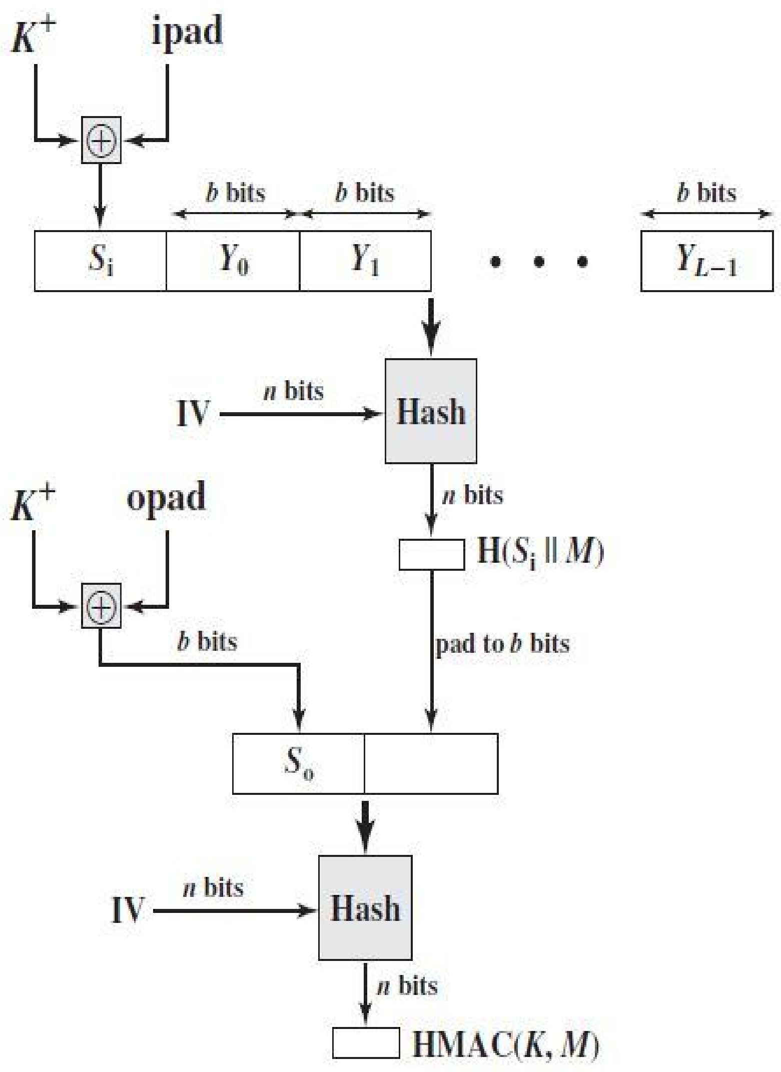
• To preserve the original performance of the hash function without incurring a significant degradation

• 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 on the embedded hash function

HMAC ALGORITHM Figure illustrates the overall operation of HMAC. The following terms are defined:

H embedded hash function (e.g., SHA-1)
M message input to HMAC (including the padding specified in the embedded hash function)
*Yi i*th block of *M*, $0 \leq i \leq (L\ 1)$
L = number of blocks in *M*
b =number of bits in a block
n =length of hash code produced by embedded hash function
K= secret key; if key length is greater than *b*, the key is input to the hash function to produce an *n*-bit key; recommended length is > *n*
K =*K*padded with zeros on the left so that the result is *b* bits in length
ipad 00110110 (36 in hexadecimal) repeated *b*/8 times
opad 01011100 (5C in hexadecimal) repeated *b*/8 times
Then HMAC can be expressed as
HMAC(*K*,*M*) H[(*K* opad) 7 H[(*K* ipad) 7 *M*]]



HMAC Structure

UNIT-II
Assignment-Cum-Tutorial Questions
SECTION-A

Objective Questions

1. Secure hash algorithm developed by___________ [ ]

a) NIST b) IEEE c)ANSI d) None of the above

2. Hash function is used to produce: [ ]

a) Finger print of a file b) Useful for message authentication

c) Both a and b d) to maintain confidentiality

3. The input block length in DES is: [ ]

a) 56 bits b) 64 bits c) 112 bits d) 128 bits

4. TDES means: [ ]

a) Triple digital encryption standard

b) Triangular data encryption standard

c) Triple data encryption standard

d) Triangular digital encryption standard

5. Which of the following is not a block cipher operating mode? [ ]

a) ECB b) CBF c) CBC d) OFB

6. Identify false statement regarding characteristics of DES [ ]

a) Each bit of the cipher text depends on all bits of the key

b) There is a statistical relationship between plaintext and cipher text

c) Avalanche effect

d) Altering a cipher text bits results in an unpredictable change

7. Disadvantages of CFB is______ [ ]

a) if data to be operated on bit or byte oriented level then only stream mode is useful

b) single error leads to errors in several blocks after the error

c) each round must wait until XOR operation finishes its scheme

d) all of the above

8. Encryption strength is based on: [ ]

a) Strength of algorithm b) Secrecy of key c) Length of key d) All of the above

9. Transposition cipher involves: [ ]

a) Replacement of blocks of text with other blocks

b) Replacement of characters of text with other character

c) Strict row to column replacement

d) Some permutation on the input text to produce cipher text

10. The secret key between members needs to be created as a ______ key when two members contact KDC. [ ]

a)public b) session c) complimentary d) private

11. The ______criterion ensures that a message cannot easily be forged[ ]
a) one-wayness

b) weak-collision-resistance

c)strong-collision resistance

d) both a and b

12. Identify which of the following is disadvantage of CBC [ ]

a) protection on order of blocks means integrity can be maintained

b) synchronization maintained automatically

c) using initial vector to randomize cipher text d) serial encryption

13. Multiplication/ Addition algorithm not uses [ ]

a) exclusive OR b) addition of integer modulo

c) multiplication of integer modulo d) subtraction of integer modulo

14. Blowfish is not suitable for smart cards because [ ]

a) it has addition operation

b) XOR operation is complex in nature

c) limited memory d) all of the above

15. Some of the parameters of Feistel cipher are [ ]

(i) Number of rounds (ii) Block size (iii) S-Box (iv) Key size

(v) Sub key Generation algorithm (vi) Function key

a) all of the above c) only (i) (ii_) (iii) (iv)

b) only (i),(ii),(iv),(v),(vi) d) only (i),(ii),(iii),(iv),(vi)

SECTION-B

SUBJECTIVE QUESTIONS

1. Explain the single round operation of DES algorithm in detail.
2. With a neat sketch explain Blowfish algorithm
3. Explain the Fiestel cipher structure with a neat sketch. And also give its significance.
4. Explain various cipher modes of operation in detail.
5. Justify the statement that the strength of DES algorithm depends on key and nonlinear S-box design.
6. Explain encryption and decryption process in TDES
7. What is Hash function? Explain different applications of cryptographic hash functions.

8. Explain the process involved in message digest generation and processing of single block in SHA-512

9. Describe the operational function of HMAC.

10. Explain various key distribution approaches.

11. What are the essential ingredients of a symmetric cipher?

INFORMATION SECURITY

UNIT III

Learning Material

Unit – III

Objectives:

- To gain the knowledge of the public-key cryptography and different public-key cryptography algorithms

Syllabus: Public-Key Cryptography

Public key cryptography principles, Public key cryptography algorithms, digital signatures, Certificate Authority and key management- Kerberos, X.509 Directory Authentication Service.

Outcomes:

Students will be able to

- Differentiate different encryption systems.
- Gain knowledge of how asymmetric encryption works
- implement symmetric encryption algorithms

Terminology Related to Asymmetric Encryption:

- Asymmetric Keys
  - Two related keys, a public key and a private key, that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification.
- Public Key Certificate
  - A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.
- Public Key (Asymmetric) Cryptographic Algorithm
  - A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.
- Public Key Infrastructure (PKI)
  - A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

3.1 Principles of public key cryptosystems

Asymmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the different keys—one a public key and one a private key. It is also known as public-key encryption.

Public-key encryption scheme has six ingredients:

- **Plaintext:** This is the readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private keys:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.

- **Cipher text:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different cipher texts.
- **Decryption algorithm:** This algorithm accepts the cipher text and the matching key and produces the original plaintext.

Conventional Vs. Public-Key Encryption

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

Requirements for Public-Key Cryptography:

1. It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_b).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:
   $C = \mathrm{E}(PU_b, M)$
3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:
   $M = \mathrm{D}(_{PRb}, C) = \mathrm{D}[PR_b, \mathrm{E}(PU_b, M)]$

4. It is computationally infeasible for an adversary, knowing the public key, PU_b, to determine the private key, PR_b.
5. It is computationally infeasible for an adversary, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.
6. The two keys can be applied in either order:
   $M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$

3.2 RSA Algorithm

- Rivest-Shamir-Adleman (RSA) scheme is the most widely accepted and implemented general-purpose approach to public-key encryption.
- The RSA scheme is a block cipher in which the plaintext and ciphertext are integers between 0 and n - 1 for some n.
- Typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024}.
- Encryption and decryption are as follows:
  - for plaintext block M and ciphertext block C,
  - $C = M^e \bmod n$
  - $M = C^d \bmod n$
  - sender and receiver must know the value of n
  - The sender knows the value of e, and only the receiver knows the value of d.
  - This is a public-key encryption algorithm with a public key of PU = {e, n} and a private key of PR = {d, n}.
  - the following requirements are must:
    - It is possible to find values of e, d, n such that $M^{ed} \bmod n = M$ for all $M < n$.
    - It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of $M < n$.
    - It is infeasible to determine d given e and n.
  -

RSA Key Setup

- Each user generates a public/private key pair by:
- select two large primes at random - p, q
- compute their system modulus n= p * q

– note ø(n) = ø(pq) = (p-1)(q-1)

- select the encryption key e
  - where $1<e<ø(n)$, gcd(e,ø(n))=1
- solve following equation to find decryption key d
  - ed=1 mod ø(n) and $0≤d≤N$
- publish their public encryption key: PU={e,n}
- keep secret private decryption key: PR={d,n}

RSA Use

- to encrypt a message M the sender:
  - obtains public key of recipient KU={e,N}
  - computes: $C=M^e$ mod N, where $0≤M<N$
- to decrypt the ciphertext C the owner:
  - uses their private key PR={d,n}
  - computes: $M=C^d$ mod N
- note that the message M must be smaller than the modulus N (block if needed)

RSA Key Generation

- users of RSA must:
  - determine two primes at random - p, q
  - select either e or d and compute the other
- primes p,q must not be easily derived from modulus N=p.q
  - means must be sufficiently large
  - typically guess and use probabilistic test
- exponents e, d are inverses, so use Inverse algorithm to compute the other

Primitive Root:

➢ A primitive root of a prime number p is one whose powers modulo generate all the integers from 1 to p-1.

- If ‘a’ is a primitive root of the prime number ‘p’, then the numbers

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

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

3.3 Diffie-Hellman Key Exchange

- 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.
- Global Public Elements
  - q prime number
  - choose α such that $\alpha < q$ and α a primitive root of q
- User A Key Generation
  - Select private X_A such that $X_A < q$
  - Calculate public Y_A $Y_A = \alpha^{(X_A)} \bmod q$
- User B Key Generation
  - Select private X_B such that $X_B < q$
  - 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
  - $K = (Y_A)^{X_B} \bmod q$

Man-in-the-Middle Attack

The Diffie-Hellman Algorithm 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_{D1} to Bob. Darth also calculates

$K_2 = (Y_A)^{X_{D2}} \bmod q$.

4. Bob receives $Y_{D1}$and calculates $K_1 = (Y_{D1})^{X_B} \bmod q$.

5. Bob transmits Y_B to Alice.

6. Darth intercepts Y_B and transmits Y_{D2} to Alice. Darth calculates

$K_1 = (Y_B)^{X_{D1}} \bmod q$.

7. Alice receives Y_{D2} and calculates $K_2 = (Y_{D2})^{X_A} \bmod q$.

The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use
of digital signatures and public-key certificates;

3.4 DIGITAL SIGNATURE

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

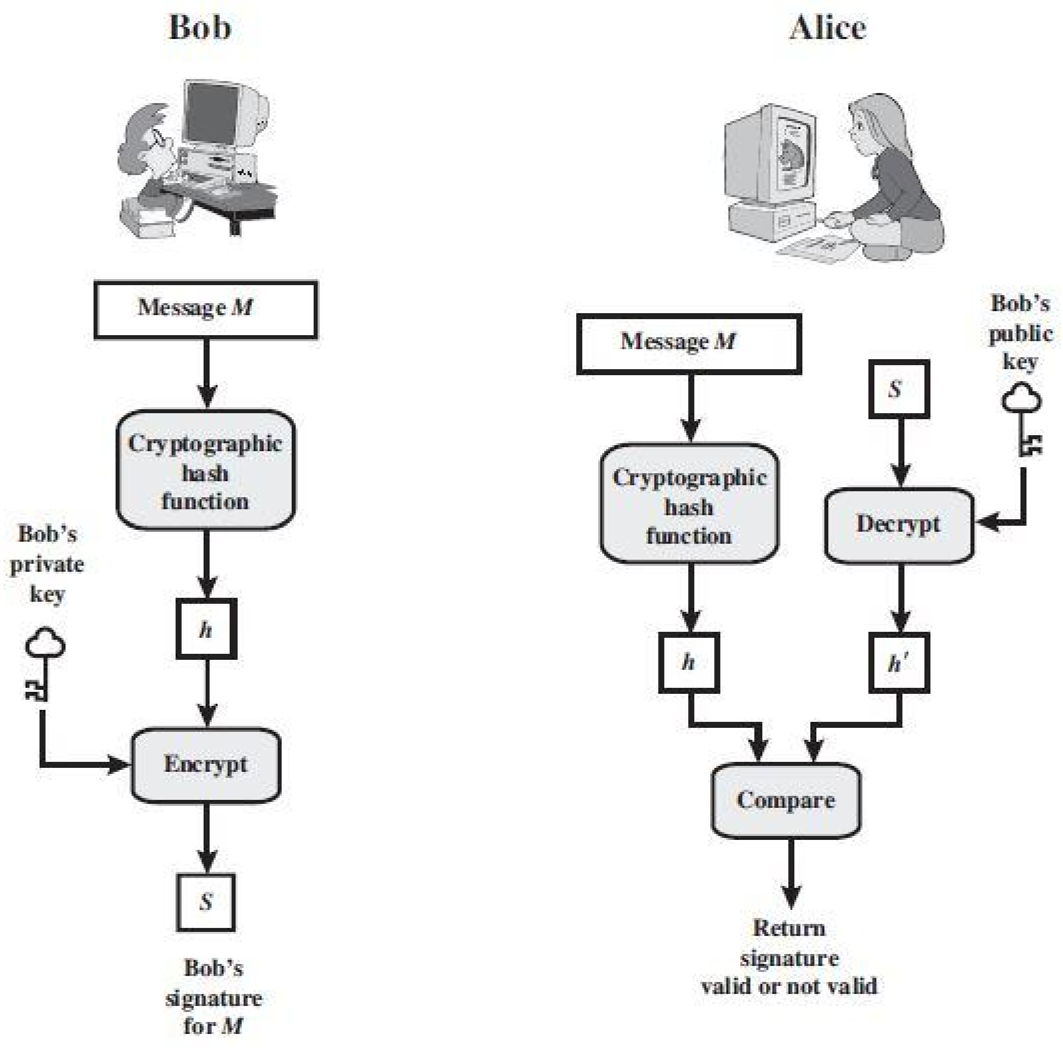
Properties

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

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

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

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



Simplified Depiction of Essential Elements of Digital Signature Process

Both scenarios are of legitimate concern. Here is an example of the first scenario:

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

- It must verify the author and the date and time of the signature.
- It must authenticate the contents at the time of the signature.

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

Thus, the digital signature function includes the authentication function.

Attacks and Forgeries [GOLD88] lists the following types of attacks, in order of increasing severity. Here A denotes the user whose signature method is being attacked, and C denotes the attacker

• Key-only attack: C only knows A's public key.

• Known message attack: C is given access to a set of messages and their signatures.

• Generic chosen message attack: C chooses a list of messages before attempting to breaks A's signature scheme, independent of A's public key. C then obtains from A valid signatures for the chosen messages.The attack is generic, because it does not depend on A's public key; the same attack is used against everyone.

• Directed chosen message attack: Similar to the generic attack, except that the list of messages to be signed is chosen after C knows A's public key but before any signatures are seen.

• Adaptive chosen message attack: C is allowed to use A as an "oracle." This means the A may request signatures of messages that depend on previously obtained message–signature pairs.

[GOLD88] then defines success at breaking a signature scheme as an outcome in which C can do any of the following with a non-negligible probability:

• Total break: C determines A's private key.

• Universal forgery: C finds an efficient signing algorithm that provides an equivalent way of constructing signatures on arbitrary messages.

• Selective forgery: C forges a signature for a particular message chosen by C.

• Existential forgery: C forges a signature for at least one message. C has no control over the message. Consequently, this forgery may only be a minor nuisance to A.

Digital Signature Requirements

On the basis of the properties and attacks just discussed, we can formulate the following requirements for a digital signature.

• The signature must be a bit pattern that depends on the message being signed.

• The signature must use some information unique to the sender to prevent both forgery and denial.

• It must be relatively easy to produce the digital signature.

• It must be relatively easy to recognize and verify the digital signature.

• It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.

• It must be practical to retain a copy of the digital signature in storage.

Direct Digital Signature

The term direct digital signature refers to a digital signature scheme that involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source.

Confidentiality can be provided by encrypting the entire message plus signature with a shared secret key (symmetric encryption). 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.

The validity of the scheme just described 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.

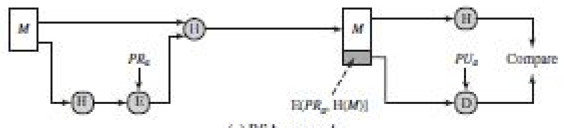
DIGITAL SIGNATURE STANDARD

The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Standard (DSS). The DSS makes use of the Secure Hash Algorithm (SHA) and presents a new digital signature technique, the Digital Signature Algorithm (DSA).The DSS was originally proposed in 1991 and revised in 1993 in response to public feedback concerning the security of the scheme. There was a further minor revision in 1996. In 2000, an expanded version of the standard was issued as FIPS 186-2, subsequently updated to FIPS 186-3 in 2009. This latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography.

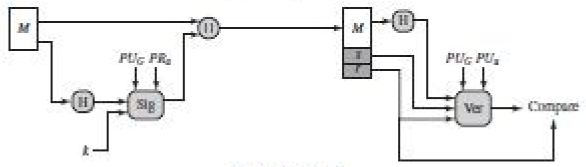
The DSS Approach

The DSS uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique. Figure contrasts the DSS approach for generating digital signatures to that used with RSA. In the RSA approach, the message to be signed is input to a hash function that produces a secure hash code of fixed length. This hash code is then encrypted using the sender's private key to form the signature. Both the message and the signature are then transmitted. The recipient takes the message and produces a hash code. The recipient also decrypts the signature using the sender's public key. If the calculated hash code matches the decrypted signature, the signature is accepted as valid. Because only the sender knows the private key, only the sender could have produced a valid signature.

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



(a) RSA approach



(b) DSS approach

At the receiving end, the hash code of the incoming message is generated. This plus the signature is input to a verification function. The verification function also depends on the global public key as well as the sender's public key(PUa) , which is paired with the sender's private key. The output of the verification function is a value that is equal to the signature component if the signature is valid.

The signature function is such that only the sender, with knowledge of the private key, could have produced the valid signature.

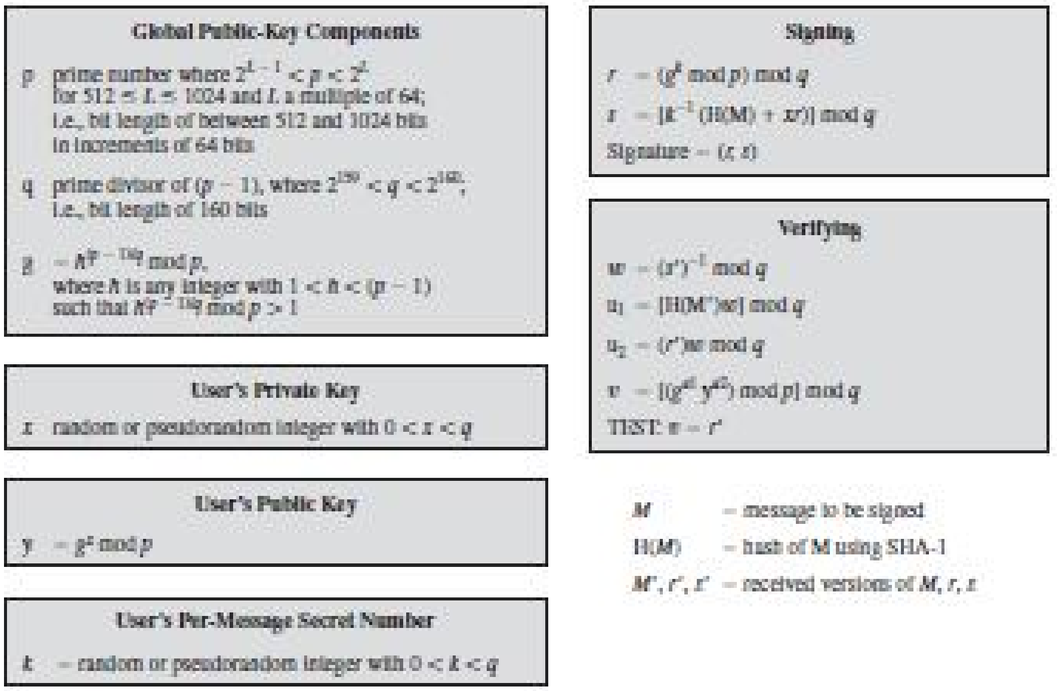
The Digital Signature Algorithm

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

Figure summarizes the algorithm. There are three parameters that are public and can be common to a group of users. A 160-bit prime number is chosen. Next, P a prime number is selected with a length between 512 and 1024 bits such that P divides (q - 1). Finally, g is chosen to be of the form h(p-1)/qmod p, where h is an integer between 1 and (q-1)with the restriction that must be greater than 12

Thus, the global public-key components of DSA have the same for as in the Schnorr signature scheme.

With these numbers in hand, each user selects a private key and generates a public key. The private key 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=gxmodp. The calculation of given is relatively straightforward. However, given the public key, it is believed to be computationally infeasible to determine ,which is the discrete logarithm of y to the base g,modp



Global Public-Key Components

p prime number where $2^{L-1} < p < 2^L$
for $512 \le L \le 1024$ and L a multiple of 64;
i.e., bit length of between 512 and 1024 bits
in increments of 64 bits

q prime divisor of $(p - 1)$, where $2^{159} < q < 2^{160}$;
i.e., bit length of 160 bits

g $= h^{(p-1)/q} \bmod p$,
where h is any integer with $1 < h < (p - 1)$
such that $h^{(p-1)/q} \bmod p > 1$

User's Private Key

x random or pseudorandom integer with $0 < x < q$

User's Public Key

$y = g^x \bmod p$

User's Per-Message Secret Number

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

Signing

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

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

Signature = (r, s)

Verifying

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

$u_1 = [H(M')w] \bmod q$

$u_2 = (r')w \bmod q$

$v = [(g^{u_1} y^{u_2}) \bmod p] \bmod q$

TEST: $v = r'$

M	= message to be signed
$H(M)$	= hash of M using SHA-1
M', r', s'	= received versions of M, r, s

To create a signature, a user calculates two quantities, and , 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 kthat should be generated
randomly or pseudorandomly and be unique for each signing. At the receiving end, verification is performed using the formulas shown in Figure The receiver generates a quantity 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 component of the signature, then the signature is validated.
Figure depicts the functions of signing and verifying.

3.5 KERBEROS

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

In particular, the following three threats exist:

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

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

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

In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access. Rather than building in elaborate authentication protocols at each server, Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users.

Two versions of Kerberos are in common use. Version 4 implementations still exist. Version 5 corrects some of the security deficiencies of version 4 and has been issued as a proposed Internet Standard

Motivation

If a set of users is provided with dedicated personal computers that have no network connections, then a user's resources and files can be protected by physically securing each personal computer. When these users instead are served by a centralized timesharing system, the time-sharing operating system must provide the security. The operating system can enforce access-control policies based on user identity and use the logon procedure to identify users.

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

1. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user identification (ID).

2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.

3. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients.

In a small, closed environment in which all systems are owned and operated by a single organization, the first or perhaps the second strategy may suffice. But in a more open environment in which network connections to other machines are supported, the third approach is needed to protect user information and resources housed at the server. Kerberos supports this third approach. Kerberos assumes a distributed client/server architecture and employs one or more Kerberos servers to provide an authentication service.

Requirements of Kerberos:

• **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.

• **Reliable:** For all services that rely on Kerberos for access control, lack of availability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture with one system able to back up another.

• **Transparent:** Ideally, the user should not be aware that authentication is taking place beyond the requirement to enter a password.

• **Scalable:** The system should be capable of supporting large numbers of clients and servers. This suggests a modular, distributed architecture.

Kerberos Version 4

Version 4 of Kerberos makes use of DES, in a rather elaborate protocol, to provide the authentication service. Viewing the protocol as a whole, it is difficult to see the need for the many elements contained therein. Therefore, we adopt a strategy used by Bill Bryant of Project Athena [BRYA88] and build up to the full protocol by looking first at several hypothetical dialogues. Each successive dialogue adds additional complexity to counter security vulnerabilities revealed in the preceding dialogue.

A SIMPLE AUTHENTICATION DIALOGUE In an unprotected network environment, any client can apply to any server for service. The obvious security risk is that of impersonation. An opponent can pretend to be another client and obtain unauthorized privileges on server machines. To counter this threat, servers must be able to confirm the identities of clients who request service. Each server can be required to undertake this task for each client/server interaction, but in an open environment, this places a substantial burden on each server. An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database. In addition, the AS shares a unique secret key with each server. These keys have been distributed physically or in some other secure manner. Consider the following hypothetical dialogue:

(1) C->AS:ID_C||P_C||ID_C

(2)AS->C:Ticket

(3)C->V: ID_C|| Ticket

Ticket = E(Kv, [IDC ||ADC ‘||IDV])

where

C= client

AS= authentication server

V=server

IDC = identifier of user on C

IDV =identifier of V

P_C =password of user on C

ADC =network address of C

Kv =secret encryption key shared by AS and V

In this scenario, the user logs on to a workstation and requests access to server V. The client module C in the user’s workstation requests the user’s password and then sends a message to the AS that includes the user’s ID, the server’s ID, and the user’s password. The AS checks its database to see if the user has supplied the proper password for this user ID and whether this user is permitted access to server V. If both tests are passed, the AS accepts the user as authentic and must now convince the server that this user is authentic. To do so, the AS creates a **ticket** that contains the user’s ID and network address and the server’s ID. This ticket is encrypted using the secret key shared by the AS and this Server. This ticket is then sent back to C. Because the ticket is encrypted, it cannot be altered by C or by an opponent.

With this ticket, C can now apply to V for service. C sends a message to V containing C’s ID and the ticket. V decrypts the ticket and verifies that the user ID in the ticket is the same as the unencrypted user ID in the message. If these two match, the server considers the user authenticated and grants the requested service.

Each of the ingredients of message (3) is significant. The ticket is encrypted to prevent alteration or forgery. The server's ID is included in the ticket so that The server can verify that it has decrypted the ticket properly. is included in the ticket to indicate that this ticket has been issued on behalf of C. Finally, serves to counter the following threat. An opponent could capture the ticket transmitted in message (2), then use the name and transmit a message of form (3) from another workstation. The server would receive a valid ticket that matches the user ID and grant access to the user on that other workstation. To prevent this attack, the AS includes in the ticket the network address from which the original request came. Now the ticket is valid only if it is transmitted from the same workstation that initially requested the ticket.

A MORE SECURE AUTHENTICATION DIALOGUE Although the foregoing scenario solves some of the problems of authentication in an open network environment, problems remain. Two in particular stand out.

First, we would like to minimize the number of times that a user has to enter a password. Suppose each ticket can be used only once. If user C logs on to a workstation in the morning and wishes to check his or her mail at a mail server, C must supply a password to get a ticket for the mail server. If C wishes to check the mail several times during the day, each attempt requires reentering the password. We can improve matters by saying that tickets are reusable. For a single logon session, the workstation can store the mail server ticket after it is received and use it on behalf of the user for multiple accesses to the mail server. However, under this scheme, it remains the case that a user would need a new ticket for every different service. If a user wished to access a print server, a mail server, a file server, and so on, the first instance of each access would require a new ticket and hence require the user to enter the password.

The second problem is that the earlier scenario involved a plaintext transmission of the password [message (1)]. An eavesdropper could capture the password and use any service accessible to the victim.

To solve these additional problems, we introduce a scheme for avoiding plaintext passwords and a new server, known as the **ticket-granting server** (TGS). The new (but still hypothetical) scenario is as follows.

Once per user logon session:

(1) C->AS: ID_C || ID_{tgs}

(2)AS->C: E(K_C,$Ticket_{tgs}$

Once per type of service:

(3)C->TGS: IDc || IDv || $Ticket_{tgs}$

(4)TGS->C:$Ticket_v$

Once per service session:

(5)C->V: ID_C || $Ticket_v$

Tickettgs = E(Ktgs, [IDC ||ADC '||IDtgs||' TS1|| Lifetime1])

Ticketv = E(Kv, [IDC ||ADC|| IDv|| TS2 || Lifetime2])

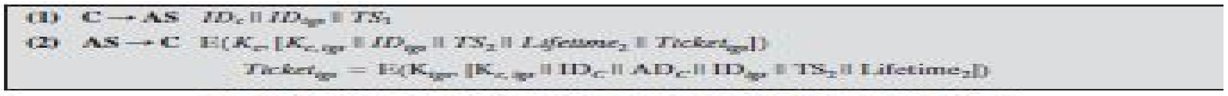
The new service, TGS, issues tickets to users who have been authenticated to AS. Thus, the user first requests a ticket-granting ticket ($Ticket_{tgs}$) from the AS. The client module in the user workstation saves this ticket. Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself. The TGS then grants a ticket for the particular service. The client saves each

service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested. Let us look at the details of this scheme:

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID to the AS, together with the TGS ID, indicating a request to use the TGS service.

2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password

(Kc), which is already stored at the AS. When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message. If the correct password is supplied, the ticket is successfully recovered.

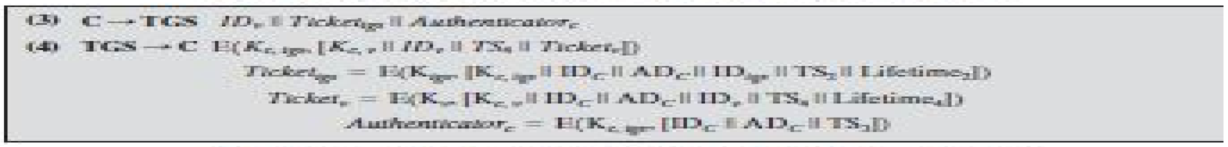


(1) $\mathbf{C} \rightarrow \mathbf{AS}$ $ID_c \parallel ID_{tgs} \parallel TS_1$

(2) $\mathbf{AS} \rightarrow \mathbf{C}$ $E(K_c, [K_{c,tgs} \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2 \parallel Ticket_{tgs}])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$

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



(3) $\mathbf{C} \rightarrow \mathbf{TGS}$ $ID_v \parallel Ticket_{tgs} \parallel Authenticator_c$

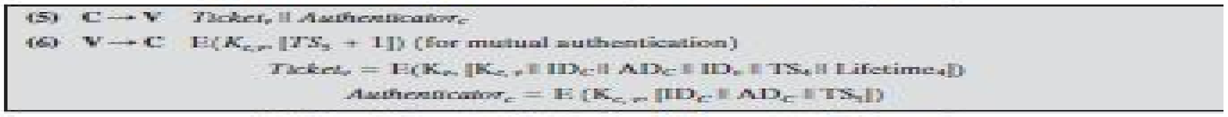
(4) $\mathbf{TGS} \rightarrow \mathbf{C}$ $E(K_{c,tgs}, [K_{c,v} \parallel ID_v \parallel TS_4 \parallel Ticket_v])$

$Ticket_{tgs} = E(K_{tgs}, [K_{c,tgs} \parallel ID_C \parallel AD_C \parallel ID_{tgs} \parallel TS_2 \parallel Lifetime_2])$

$Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$

$Authenticator_c = E(K_{c,tgs}, [ID_C \parallel AD_C \parallel TS_3])$

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



(5) $\mathbf{C} \rightarrow \mathbf{V}$ $Ticket_v \parallel Authenticator_c$

(6) $\mathbf{V} \rightarrow \mathbf{C}$ $E(K_{c,v}, [TS_5 + 1])$ (for mutual authentication)

$Ticket_v = E(K_v, [K_{c,v} \parallel ID_C \parallel AD_C \parallel ID_v \parallel TS_4 \parallel Lifetime_4])$

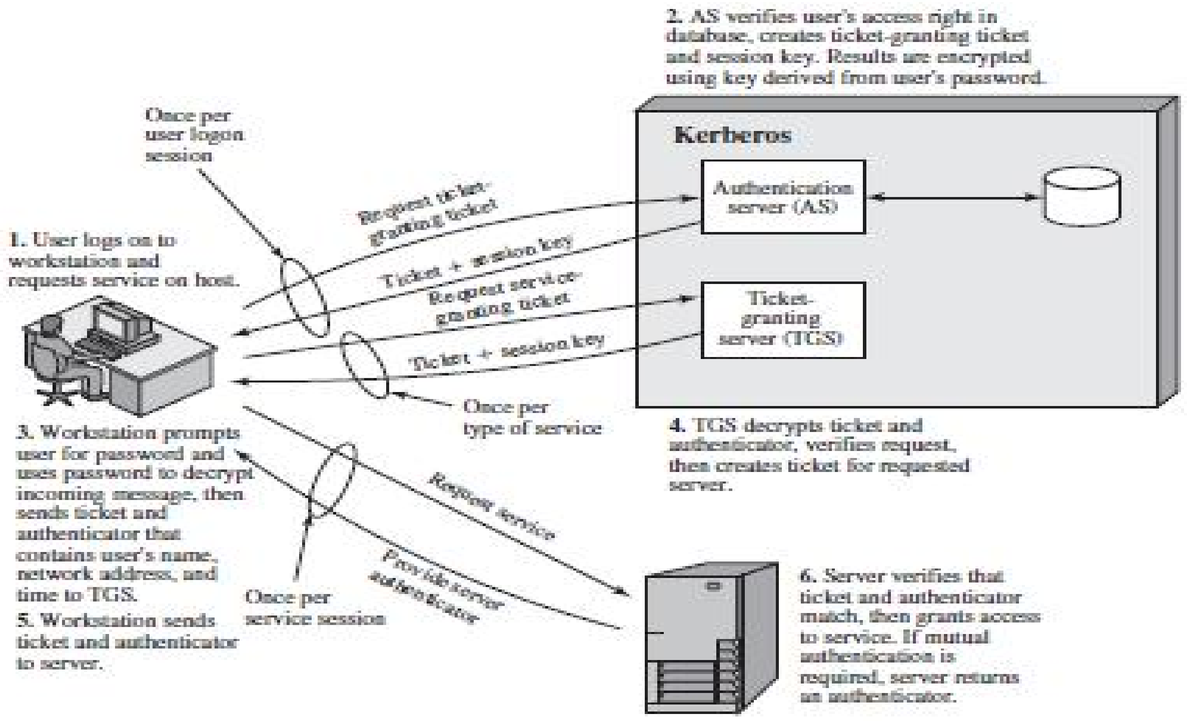
$Authenticator_c = E(K_{c,v}, [ID_C \parallel AD_C \parallel TS_5])$

(c) Client/Server Authentication Exchange to obtain service

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

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.

2. The Kerberos server must share a secret key with each server.All servers are registered with the Kerberos server.


2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
Kerberos
Authentication server (AS)
Ticket-granting server (TGS)
Once per user logon session
Request ticket-granting ticket
Ticket + session key
Request service-granting ticket
Ticket + session key
1. User logs on to workstation and requests service on host.
Once per type of service
3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
Request service
Provide server authenticator
Once per service session
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Such an environment is referred to as a **Kerberos realm**. The concept of **realm** can be explained as follows. A Kerberos realm is a set of managed nodes that share the same Kerberos database. The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room. A read-only copy of the Kerberos database might also reside on other

Kerberos computer systems. However, all changes to the database must be made on the master computer system. Changing or accessing the contents of a Kerberos database requires the Kerberos master password. A related concept is that of a **Kerberos principal**, which is a service or user that is known to the Kerberos system. Each Kerberos principal is identified by its principal name.

Principal names consist of three parts: a service or user name, an instance name, and a realm name Networks of clients and servers under different administrative organizations typically constitute different realms. That is, it generally is not practical or does not conform to administrative policy to have users and servers in one administrative domain registered with a Kerberos server elsewhere. However,users in one realm may need access to servers in other realms, and some servers may be willing to provide service to users from other realms, provided that those users are authenticated.

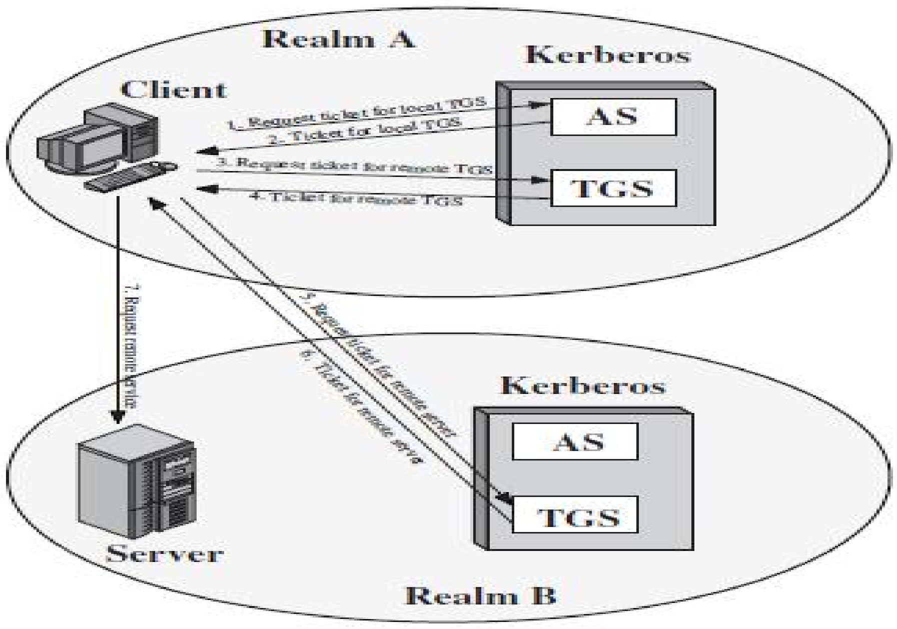
Kerberos provides a mechanism for supporting such interrealm authentication. For two realms to support interrealm authentication, a third requirement is added:

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

With these ground rules in place, we can describe the mechanism as follows (Figure 2): A user wishing service on a server in another realm needs a ticket for that server. The user's client follows the usual procedures to gain access to the local TGS and then requests a ticket-granting ticket for a remote TGS (TGS in another realm).The client can then apply to the remote TGS for a service-granting ticket for

the desired server in the realm of the remote TGS.

The details of the exchanges illustrated in Figure 2 are as follows


Realm A
Kerberos
Client
AS
TGS
1. Request ticket for local TGS
2. Ticket for local TGS
3. Request ticket for remote TGS
4. Ticket for remote TGS
7. Request remote service
5. Request ticket for remote server
6. Ticket for remote server
Kerberos
AS
TGS
Server
Realm B

(1) $C \rightarrow AS$: $ID_c \| ID_{tgs} \| TS_1$

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

(3) $C \rightarrow TGS$: $ID_{tgsrem} \| Ticket_{tgs} \| Authenticator_c$

(4) $TGS \rightarrow C$: $E(K_{c,tgs}, [K_{c,tgsrem} \| ID_{tgsrem} \| TS_4 \| Ticket_{tgsrem}])$

(5) $C \rightarrow TGS_{rem}$: $ID_{vrem} \| Ticket_{tgsrem} \| Authenticator_c$

(6) $TGS_{rem} \rightarrow C$: $E(K_{c,tgsrem}, [K_{c,vrem} \| ID_{vrem} \| TS_6 \| Ticket_{vrem}])$

(7) $C \rightarrow V_{rem}$: $Ticket_{vrem} \| Authenticator_c$

The ticket presented to the remote server (Vrem) indicates the realm in which the user was originally authenticated. The server chooses whether to honor the remote request.

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

This led to the following **environmental shortcomings**.

1. Encryption system dependence: Version 4 requires the use of DES. Export restriction on DES as well as doubts about the strength of DES were thus of concern. In version 5, ciphertext is tagged with an encryption-type identifier so that any encryption technique may be used. Encryption keys are tagged

with a type and a length, allowing the same key to be used in different algorithms and allowing the specification of different variations on a given

algorithm.

2. Internet protocol dependence: Version 4 requires the use of Internet Protocol (IP) addresses. Other address types, such as the ISO network address, are not accommodated. Version 5 network addresses are tagged with type and length, allowing any network address type to be used.

3. Message byte ordering: In version 4, the sender of a message employs a byte ordering of its own choosing and tags the message to indicate least significant byte in lowest address or most significant byte in lowest address..

4. Ticket lifetime: Lifetime values in version 4 are encoded in an 8-bit quantity in units of five minutes. Thus, the maximum lifetime that can be expressed is 2^8*5=1280 minutes (a little over 21 hours).This may be inadequate for some applications   In version 5, tickets include an explicit start time and

end time, allowing tickets with arbitrary lifetimes.

5. Authentication forwarding: Version 4 does not allow credentials issued to one client to be forwarded to some other host and used by some other client. This capability would enable a client to access a server and have that server access another server on behalf of the client. For example, a client issues a request to a print server that then accesses the client's file from a file server, using the client's

credentials for access.Version 5 provides this capability.

6. Interrealm authentication: In version 4, interoperability among realms requires on the order of Kerberos-to-Kerberos relationships, as described earlier. Version 5 supports a method that requires fewer relationships, as described shortly.

Apart from these environmental limitations, there are **technical deficiencies** in the version 4 protocol itself. Most of these deficiencies and version 5 attempts to address these. The deficiencies are the following.

1. Double encryption: Note in Table.1 [messages (2) and (4)] that tickets provided to clients are encrypted twice—once with the secret key of the target server and then again with a secret key known to the client. The second encryption is not necessary and is computationally wasteful.

2. PCBC encryption: Encryption in version 4 makes use of a nonstandard mode of DES known as **propagating cipher block chaining** (PCBC).9 It has been demonstrated that this mode is vulnerable to an attack involving the interchange of ciphertext blocks . PCBC was intended to provide an integrity check as part of the encryption operation. Version 5 provides explicit integrity mechanisms, allowing the standard CBC mode to be used for encryption. In particular, a checksum or hash code is attached to the message prior to encryption using CBC.

3. Session keys: Each ticket includes a session key that is used by the client to encrypt the authenticator sent to the service associated with that ticket. In addition, the session key may subsequently be used by the client and the server to protect messages passed during that session.

However, because the same ticket may be used repeatedly to gain service from a particular server, there is the risk that an opponent will replay messages from an old session to the client or the server. In version 5, it is possible for a client and server to negotiate a subsession key, which is to be used only for that one connection. A new access by the client would result in the use of a new subsession key.

4. Password attacks: Both versions are vulnerable to a password attack. The message from the AS to the client includes material encrypted with a key based on the client's password.10 An opponent can capture this message and attempt to decrypt it by trying various passwords. If the result of a test decryption is of the proper form, then the opponent has discovered the client's password and may subsequently use it to gain authentication credentials from Kerberos.

3.6 X.509 CERTIFICATES

ITU-T recommendation X.509 is part of the X.500 series of recommendations that define a directory service. The directory is, in effect, a server or distributed set of servers that maintains a database of information about users. The information includes a mapping from user name to network address, as well as other attributes and information about the users. X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates.. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.

X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety of contexts. For example, the X.509 certificate format is used in S/MIME, IP Security and SSL/TLS

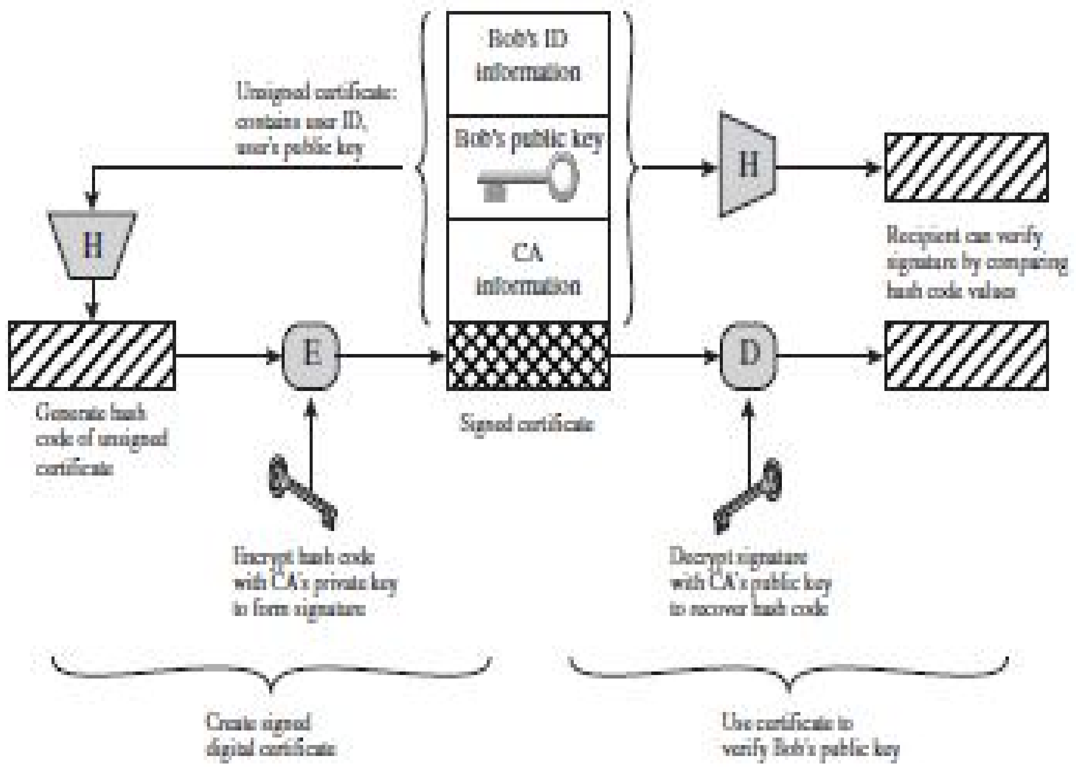
X.509 was initially issued in 1988. The standard was subsequently revised to address some of the security concerns documented in [IANS90] and [MITC90]; a revised recommendation was issued in 1993. A third version was issued in 1995 and revised in 2000.

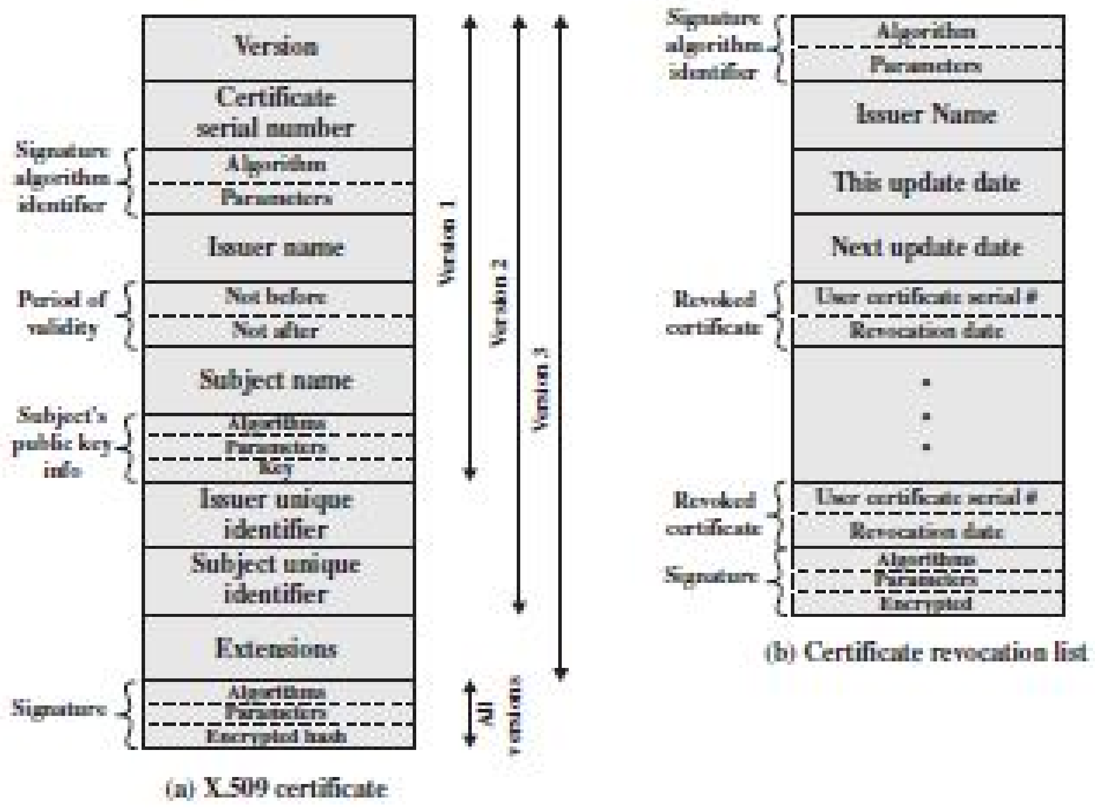
X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends RSA. The digital signature scheme is assumed to require the use of a hash function. Again, the standard does not dictate a specific hash algorithm. The 1988 recommendation included the description of a recommended hash algorithm; this algorithm has since been shown to be insecure and was dropped from the 1993 recommendation. Figure illustrates the generation of a public-key certificate.

Certificates

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

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


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



X.509 Formats

• **Version:** Differentiates among successive versions of the certificate format; the default is version 1. If the *issuer unique identifier* or *subject unique identifier* are present, the value must be version 2. If one or more extensions are present, the version must be version 3.

• **Serial number:** An integer value unique within the issuing CA that is unambiguously associated with this certificate.

• **Signature algorithm identifier:** The algorithm used to sign the certificate together with any associated parameters. Because this information is repeated in the signature field at the end of the certificate, this field has little, if any, utility.

• **Issuer name:** X.500 is the name of the CA that created and signed this certificate.

• **Period of validity:** Consists of two dates: the first and last on which the certificate is valid.

• **Subject name:** The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.

• **Subject's public-key information:** The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

• **Issuer unique identifier:** An optional-bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.

• **Subject unique identifier:** An optional-bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.

• **Extensions:** A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.

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

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

The standard uses the following notation to define a certificate:

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

Where

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

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

V=version of the certificate

SN=serial number of the certificate

AI=identifier of the algorithm used to sign the certificate

CA=name of certificate authority

UCA=optional unique identifier of the CA

A=name of user A

Ap=optional unique identifier of the user A public key of user A

T^A =period of validity of the certificate

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

OBTAINING A USER’S CERTIFICATE User certificates generated by a CA have the following characteristics:

• Any user with access to the public key of the CA can verify the user public key

that was certified.

• No party other than the certification authority can modify the certificate without

this being detected.

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

If all users subscribe to the same CA, then there is a common trust of that CA.

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

If there is a large community of users, it may not be practical for all users to subscribe to the same CA. Because it is the CA that signs certificates, each participating user must have a copy of the CA’s own public key to verify signatures. This public key must be provided to each user in an absolutely secure (with respect to integrity and authenticity) way so that the user has confidence in the associated certificates. Thus, with many users, it may be more practical for there to be a number of CAs, each of which securely provides its public key to some fraction of the users.

Now suppose that A has obtained a certificate from certification authority X_1 and B has obtained a certificate from CA X_2 . If A does not securely know the public key of X_2 , then B’s certificate, issued by X_2 , is useless to A.A can read B’s certificate, but A cannot verify the signature.

However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key.

Step 1 A obtains from the directory the certificate of signed by B . Because A securely knows 's public key, A can obtain 's public key from its certificate and verify it by means of 's signature on the certificate.

Step 2 A then goes back to the directory and obtains the certificate of B signed by X2. Because A now has a trusted copy of X2's public key, A can verify the signature and securely obtain B's public key.

A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as

X1<<X2>>X2<<B>>

In the same fashion, B can obtain A's public key with the reverse chain:

X2<<X1>>X1<<A>>

This scheme need not be limited to a chain of two certificates. An arbitrarily long path of CAs can be followed to produce a chain. A chain with N elements would be expressed as

X1<<X2>>X2<<X3>>….XN<<B>>

In this case, each pair of CAs in the chain (X_i, X_{i+1})must have created certificates for each other.

All these certificates of CAs by CAs need to appear in the directory, and the user needs to know how they are linked to follow a path to another user's publickey certificate. X.509 suggests that CAs be arranged in a hierarchy so that navigation is straightforward.

UNIT-III
Assignment-Cum-Tutorial Questions
SECTION-A

Objective Questions

1. _________ encryption algorithms is known as public key cryptography
a) Symmetric key cryptography c) private key cryptography
b) Single key cryptography d) Asymmetric key cryptography

2. The ________ attack can endanger the security of the Diffie-Hellman method if two parties are not authenticated to each other. [ ]
a) man-in-the-middle b) ciphertext attack
c) plaintext attack d) none of the above

3. One of major drawback of conventional encryption is_________ [ ]

a) integrity b) forgery c) encryption d) decryption

4. Public key ring is _______ [ ]

a) a ring topology at source

b) a ring topology at destination

c) set of public keys available at source or destination

d) ring topology that completes network with source and destination hosts

5. if public key is used for decryption then this model provides_____[ ]

a) authentication b) confidentiality

c) both a) and b) d) neither a) nor b)

6. Which of the following is not an attack on RSA algorithm are ____[ ]

a) brute force attacks b) timing attacks

c) mathematical attacks d) magnitude attacks

7. In public key certificates signature can be verified by anyone who knows the public key of _______ [ ]

a) source b)Destination c) private key d) certicate authority

8. Kerberos makes use of ________ algorithm [ ]

a) RSA b) DES c) BLOWFISH d) IDEA

9. Which of the following is not involoved in simple authentication dialogue of Kerberos version4 [ ]

a) client b) server c) authentication server d) key distribution server

10. Secrecy and authentication both are maintained when [ ]
a) Data encrypted with private key
b) Data encrypted with public key
c) Data encrypted with private and public keys
d) Data is decrypted with encrypted key
11. Applications of Asymmetric key cryptography are [ ]
a) Encryption and decryption
b) Signing and verifying documents , digital signatures
c) Key exchange purpose
d) All of the above
12. Identify wrong possible attack for public key cryptography [ ]
a) Brute force attack c) Public key finding
b) Private key generation using pubic key d) Probable message attack
13. RSA algorithm is not based on____________ [ ]
a) Exponentials c) Modulus numbers
b) Prime numbers d) Ring numbers
14. Consider M is message, e is public key, n is modulo, d is private key then formula for encryption in RSA algorithm to generate cipher text C is_________ [ ]
a) $C=M^e$ mod n b) $C=M^n$ mod e c) $C=d * M^e$ mod n d) $C=n * M^e$ mod n
15. Purpose of Authentication Service exchange is [ ]
a) Used to obtain service-granting ticket
b) Used to obtain ticket-granting ticket
c) Used to obtain granting-service-ticket –exchange ticket
d) Used to obtain service
16. Lifetime in a ticket refers to________ [ ]
a) Length of time for which service is valid
b) Length of time for which ticket is valid
c) Indication of authenticated server
d) Indication of authenticated ticket
17. A<<B>> in X.509 indicates that________ [ ]
a) The certificate of user A issued by Certification Authority B
b) The certificate of user B issued by Certification Authority A

c) The certificate of user A issued by user B
d) The certificate of server A issued by user B

SECTION-B

SUBJECTIVE QUESTIONS

1. Discuss Diffie –Hellman Key exchange protocols in detail
2. Write and explain Client/ Server Authentication Exchange service in Kerberos
   version 4.
3. Explain the RSA algorithm
4. Explain in detail Digital Signature Standard approach and its algorithm
5. Give the format for X.509 certificate. How are users certificates obtained? [8] b) Explain the authentication services provided by X.509.
6. In detail explain different possible approaches for attacking RSA algorithm
7. What is the purpose of digital signature? Explain its properties and requirements.
8. With a neat sketch explain overview of Message Exchanges in Kerberos version 5.
9. What is the purpose of the X.509 standard?
10. What problem was Kerberos designed to address?
11. Consider the following scheme
    (a) Pick an odd number E
    (b) Pick two prime numbers, P and Q, where (P-1)(q-1)-1 is evenly distributed by E
    (c) Multiply P and Q to get N
    (d) Calculate D=((P-1)(Q-1)(E-1)+1)/E
    Is this scheme is equivalent to RSa? Show why or why not
12. Perform encryption and decryption using RSA algorithm for the following
    (i) P=3, q=11, d=7, M=5
    (ii) P=11,q=13, e=11,M=7
13. Compute cipher text for M=88, p=17 and q=11 using RSA algorithm
14. Perform encryption and decryption using the RSA algorithm P = 3, q= 11,e = 7, M = 5.
15. Consider a diffie hellman scheme with a common prime q=11 and a primitive root a=2
    (i) If user A has public key YA=9, what is A's private key Xa?

(ii) If user B has public key Y, B=3, what is the shared key K?

16. Users A and B use Diffie-Hellman key exchange scheme using prime q=71 and primitive root α =2.
a) User A has private key Xa=5, What is A's public key Ya?
b) User B has private key Xb=12, what is B's public key Yb?
c) What is the shared secret key?

SECTION-C

QUESTIONS AT THE LEVEL OF GATE

1. Using public key cryptography, X adds a digital signature to message M, encrypts , and sends it to Y, where it is decrypted. Which one of the following sequences of keys is used for the operations? [ ]
(a) Encryption: X's private key followed by Y's private key; Decryption: X's public key followed by Y's public key
(b) Encryption: X's private key followed by Y's public key; Decryption: X's public key followed by Y's private key
(c) Encryption: X's public key followed by Y's private key; Decryption: Y's public key followed by X's private key
(d) Encryption: X's private key followed by Y's public key; Decryption: Y's private key followed by X's public key **[GATE 2013]**

2. In the RSA public key cryptosystem, the private and public keys are (e, n) and (d, n) respectively, where n = p*q and p and q are large primes. Besides, n is public and p and q are private. Let M be an integer such that 0 < M < n and f(n) = (p- 1)(q-1). Now consider the following equations.

I. $M' = M^e \bmod n$
$M = (M')^d \bmod n$

II. $ed \equiv 1 \bmod n$

III. $ed \equiv 1 \bmod f(n)$

IV. $M' = M^e \bmod f(n)$

$M = (M')^d \bmod f(n)$

Which of the above equations correctly represent RSA cryptosystem?

(a) I and II [ ]
(b) I and III
(c) II and IV
(d) III and IV **[GATE 2009]**

3. A sender is employing public key cryptography to send a secret message to a receiver. Which one of the following statements is TRUE? [ ]
(a) Sender encrypts using receiver's public key
(b) Sender encrypts using his own public key
(c) Receiver decrypts using sender's public key
(d) Receiver decrypts using his own public key **[GATE IT 2004]**

4. The total number of keys required for a set of n individuals to be able to communicate with each other using secret key and public key crypto-systems, respectively are: [ ]
(a) n(n-1) and 2n
(b) 2n and ((n(n – 1))/2)
(c) ((n(n – 1))/2) and 2n
(d) ((n(n – 1))/2) and n **[GATE IT 2008]**

INFORMATION SECURITY

UNIT IV
Learning Material

Unit – IV

Objectives:

Syllabus: Authentication applications & Introduction to IP Security

Email privacy- Pretty Good Privacy (PGP) and S/MIME Overview, IP Security Architecture, Authentication Header, Encapsulating Security Payload, Combining Security Associations and Key Management

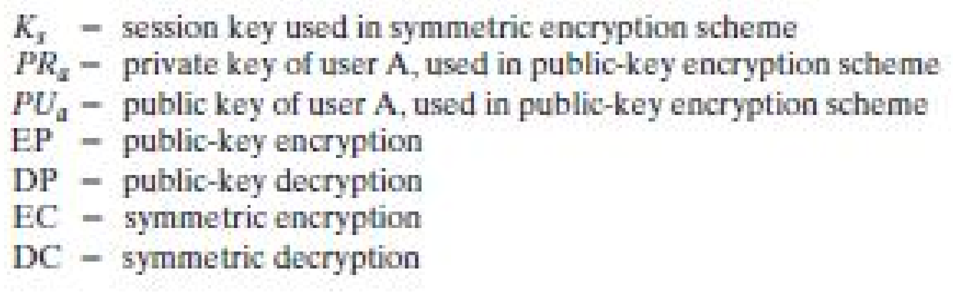
PRETTY GOOD PRIVACY

PGP is a remarkable phenomenon. Largely the effort of a single person, Phil Zimmermann, PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. PGP has grown explosively and is now widely used. A number of reasons can be cited for this growth.

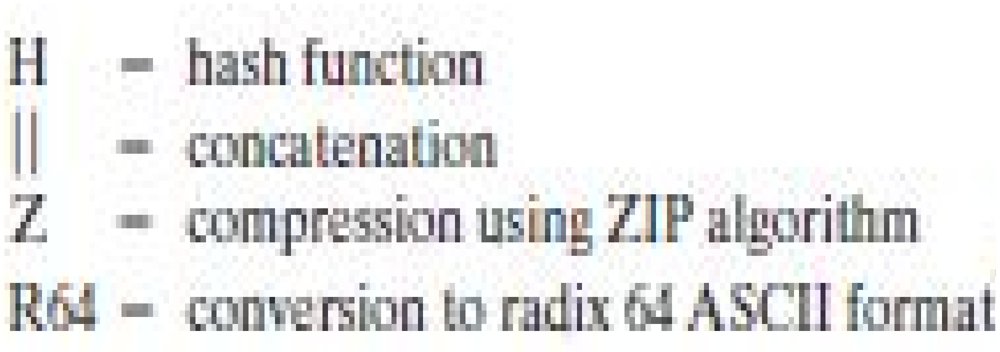
1. It is available free worldwide in versions that run on a variety of platforms, including Windows, UNIX, Macintosh, and many more. In addition, the commercial version satisfies users who want a product that comes with vendor support.
2. It is based on algorithms that have survived extensive public review and are considered extremely secure. Specifically, the package includes RSA, DSS, and Diffie-Hellman for public-key encryption;CAST-128, IDEA, and 3DES for symmetric encryption; and SHA-1 for hash coding.
3. It has a wide range of applicability, from corporations that wish to select and enforce a standardized scheme for encrypting files and messages to individuals who wish to communicate securely with others worldwide over the Internet and other networks.
4. It was not developed by, nor is it controlled by, any governmental or standards organization. For those with an instinctive distrust of "the establishment," this makes PGP attractive.
5. PGP is now on an Internet standards track (RFC 3156; MIME Security with OpenPGP). Nevertheless, PGP still has an aura of an antiestablishment endeavor.

Notation

The following symbols are used.



K_s	=	session key used in symmetric encryption scheme
PR_a	=	private key of user A, used in public-key encryption scheme
PU_a	=	public key of user A, used in public-key encryption scheme
EP	=	public-key encryption
DP	=	public-key decryption
EC	=	symmetric encryption
DC	=	symmetric decryption



H	-	hash function
||	-	concatenation
Z	-	compression using ZIP algorithm
R64	-	conversion to radix 64 ASCII format

Operational Description:
The actual operation of PGP, as opposed to the management of keys, consists of five services: authentication, confidentiality, compression, e-mail compatibility And segmentation and reassembly

AUTHENTICATION: Figure 1a illustrates the digital signature service provided by PGP. This is the digital signature scheme.The sequence is as follows.

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

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

As always, one must address the problem of key distribution. In PGP, each symmetric key is used only once. That is, a new key is generated as a random 128-bit number for each message. Thus, although this is referred to in the documentation as a session key, it is in reality a one-time key. Because it is to be used only once, the session key is bound to the message and transmitted with it.To protect the key, it is encrypted with the receiver’s public key. Figure b illustrates the sequence, which can be described as follows.

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

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

CONFIDENTIALITY AND AUTHENTICATION: As Figure 1c illustrates, both services may be used for the same message. First, a signature is generated for the plaintext message and prepended to the message. hen the plaintext message plus signature is encrypted using CAST-128 (or IDEA or 3DES), and the session key is encrypted using RSA This sequence is preferable to the opposite: encrypting the message and then generating a signature for the encrypted message. It is generally more convenient to store a signature with a plaintext version of a message. Furthermore, for purposes of third-party verification, if the signature is performed first, a third party need not be concerned with the symmetric key when verifying the signature. In summary, when both services are used, the sender first signs the message with its own private key, then encrypts the message with a session key, and finally encrypts the session key with the recipient's public key.

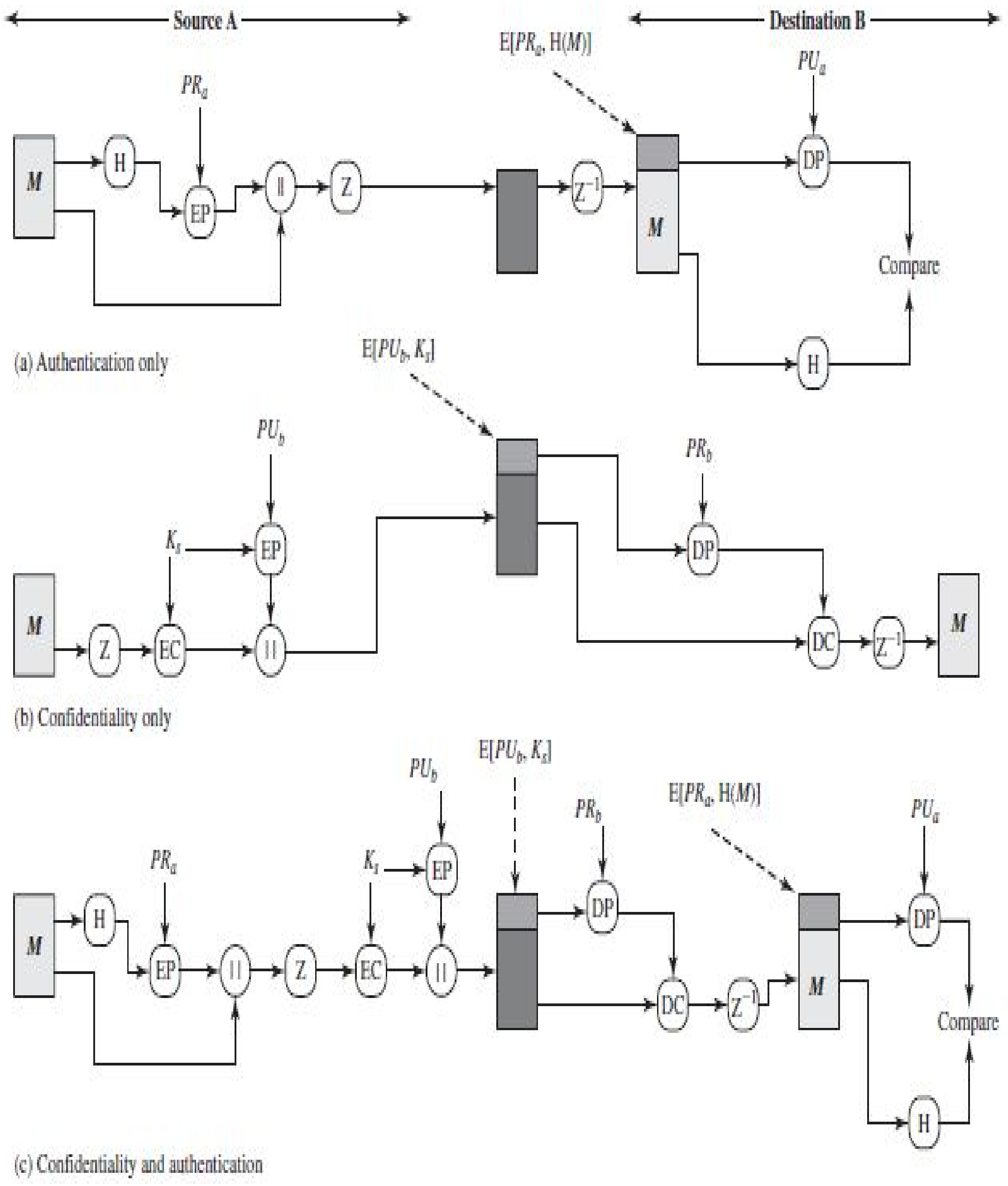


FIGURE:1

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

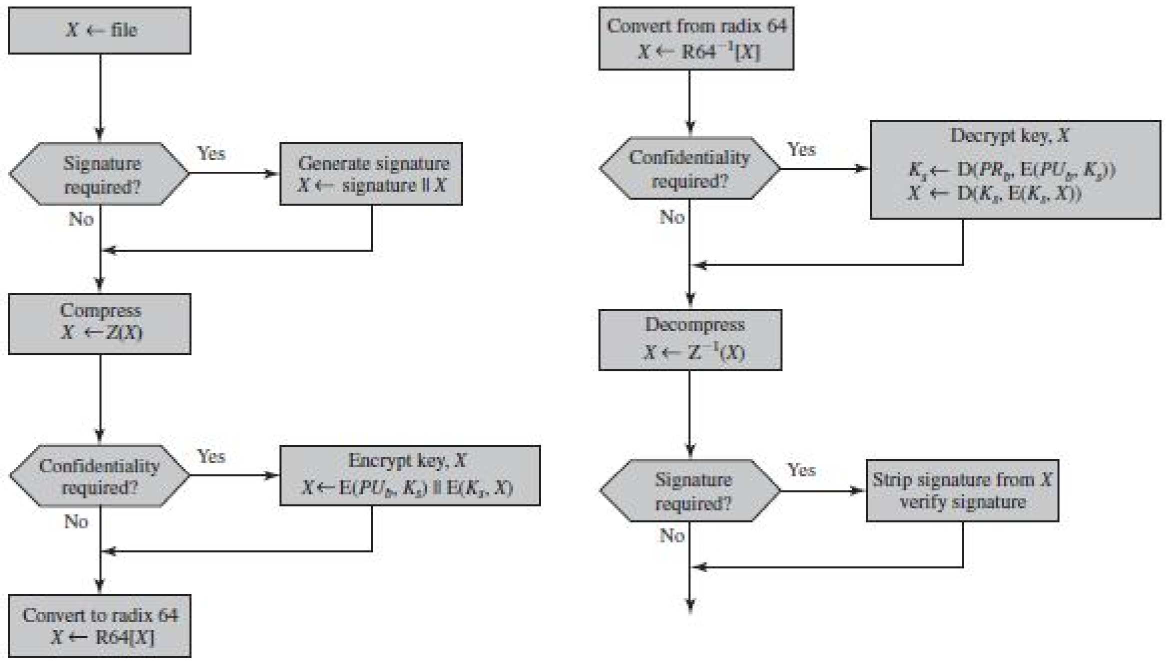
The placement of the compression algorithm, indicated by Z for compression and Z–1 for decompression in Figure 1, is critical.
1. The signature is generated before compression for two reasons:

a. It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.
b. Even if one were willing to generate dynamically a recompressed message for verification, PGP's compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and, as a result, produce different compressed forms. However, these different compression algorithms are interoperable because any version of the algorithm can correctly decompress the output of any other version. Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm.
2. Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.

E-MAIL COMPATIBILITY :When PGP is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender's private key). If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key). Thus, part or all of the resulting block consists of a stream of arbitrary 8-bit octets.
However, many electronic mail systems only permit the use of blocks consisting of ASCII text. To accommodate this restriction, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters. The scheme used for this purpose is radix-64 conversion. Each group of three octets of binary data is mapped into four ASCII characters. This format also appends a CRC to detect transmission errors. The use of radix 64 expands a message by 33%. Fortunately, the session key and signature portions of the message are relatively compact, and the plaintext message has been compressed.

In fact, the compression should be more than enough to compensate for the radix-64 expansion.

Figure 2 shows the relationship among the four services so far discussed. On transmission (if it is required), a signature is generated using a hash code of the uncompressed plaintext. Then the plaintext (plus signature if present) is compressed. Next, if confidentiality is required, the block (compressed plaintext or compressed signature plus plaintext) is encrypted and prepended with the public-keyencrypted symmetric encryption key. Finally, the entire block is converted to radix-64 format. On reception, the incoming block is first converted back from radix-64 format to binary. Then, if the message is encrypted, the recipient recovers the session key and decrypts the message. The resulting block is then decompressed. If the message is signed, the recipient recovers the transmitted hash code and compares it to its own calculation of the hash code.

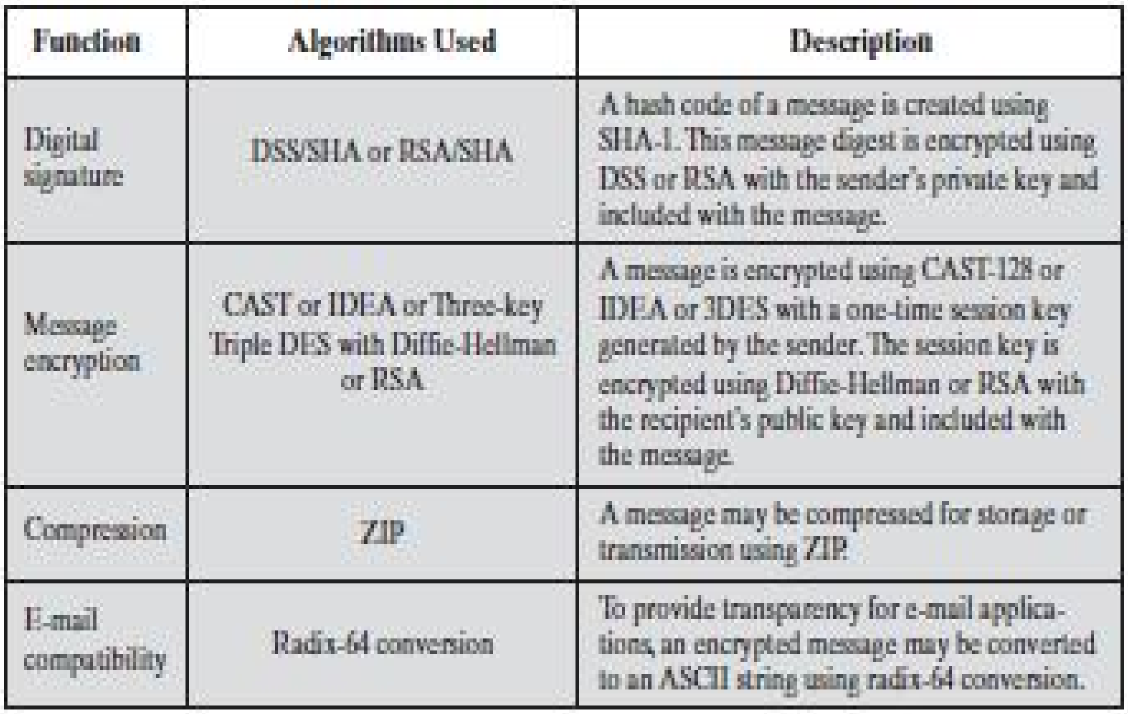

$X \leftarrow$ file
Signature required?
Yes
No
Generate signature $X \leftarrow$ signature $\|$ X
Compress $X \leftarrow Z(X)$
Confidentiality required?
Yes
No
Encrypt key, X $X \leftarrow E(PU_b, K_s) \| E(K_s, X)$
Convert to radix 64 $X \leftarrow R64[X]$
Convert from radix 64 $X \leftarrow R64^{-1}[X]$
Confidentiality required?
Yes
No
Decrypt key, X $K_s \leftarrow D(PR_b, E(PU_b, K_s))$ $X \leftarrow D(K_s, E(K_s, X))$
Decompress $X \leftarrow Z^{-1}(X)$
Signature required?
Yes
No
Strip signature from X verify signature

(a) Generic transmission diagram (from A) (b) Generic reception diagram (to B)

Segmentation and Reassembly: E-mail facilities often are restricted to a maximum message length. For example, many of the facilities accessible through the Internet impose a maximum length of 50,000 octets. Any message longer than that must be broken up into smaller segments, each of which is mailed separately. To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to send via e-mail.

The segmentation is done after all of the other processing, including the radix-64 conversion. Thus, the session key component and signature component appear only once, at the beginning of the first segment.

Summary of the operations:

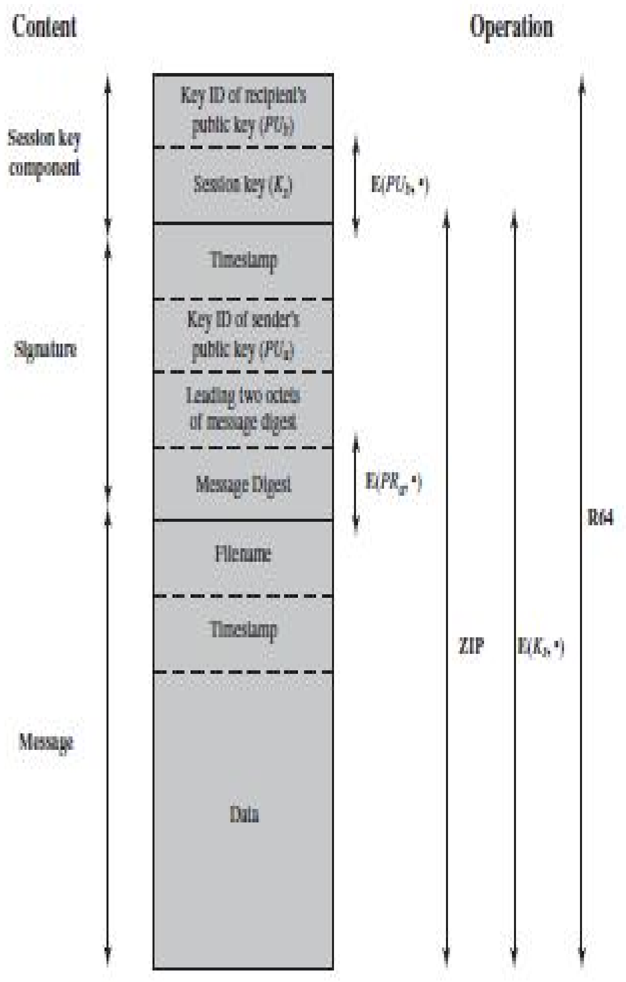


Function	Algorithms Used	Description
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman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

PGP Message format:

The message component includes the actual data to be stored or transmitted, as well as a filename and a timestamp that specifies the time of creation.
The signature component includes the following.
• **Timestamp:** The time at which the signature was made.
• **Message digest:** The 160-bit SHA-1 digest encrypted with the sender's private signature key. The digest is calculated over the signature timestamp concatenated with the data portion of the message component. The inclusion of the signature timestamp in the digest insures against replay types of attacks. The exclusion of the filename and timestamp portions of the message component ensures that detached signatures are exactly the same as attached signatures prefixed to the message. Detached signatures are calculated on a separate file that has none of the message component header fields.

• **Leading two octets of message digest:** Enables the recipient to determine if the correct public key was used to decrypt the message digest for authentication by comparing this plaintext copy of the first two octets with the first two octets of the decrypted digest. These octets also serve as a 16-bit frame check sequence for the message.
• **Key ID of sender's public key:** Identifies the public key that should be used to decrypt the message digest and, hence, identifies the private key that was used to encrypt the message digest. The message component and optional signature component may be compressed using ZIP and may be encrypted using a session key. The session key component includes the session key and the identifier of the recipient's public key that was used by the sender to encrypt the session key. The entire block is usually encoded with radix-64 encoding.



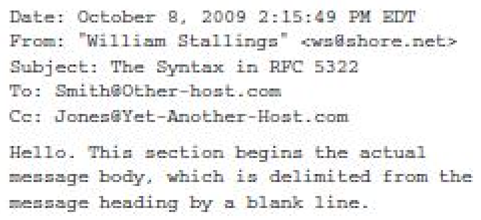
Notation:

E(PU_b, •)	= encryption with user b's public key
E(PR_a, •)	= encryption with user a's private key
E(K_s, •)	= encryption with session key
ZIP	= Zip compression function
R64	= Radix-64 conversion function

4.2 S/MIME:

Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security. To understand S/MIME, we need first to have a general understanding of the underlying e-mail format that it uses, namely MIME. But to understand the significance of MIME, we need to go back to the traditional e-mail

format standard, RFC 822, which is still in common use. The most recent version of this format specification is RFC 5322 (Internet Message Format).
RFC 5322
RFC 5322 defines a format for text messages that are sent using electronic mail. It has been the standard for Internet-based text mail messages and remains in common use. In the RFC 5322 context, messages are viewed as having an envelope and contents. The envelope contains whatever information is needed to accomplish transmission and delivery. The contents compose the object to be delivered to the recipient. The RFC 5322 standard applies only to the contents. However, the content standard includes a set of header fields that may be used by the mail system to create the envelope, and the standard is intended to facilitate the acquisition of such information by programs. The overall structure of a message that conforms to RFC 5322 is very simple. A message consists of some number of header lines (the header) followed by unrestricted text (the body). The header is separated from the body by a blank line. Put differently, a message is ASCII text, and all lines up to the first blank line are assumed to be header lines used by the user agent part of the mail system.
A header line usually consists of a keyword, followed by a colon, followed by the keyword's arguments; the format allows a long line to be broken up into several lines. The most frequently used keywords are From, To, Subject, and Date. Here is an example message:



```
Date: October 8, 2009 2:15:49 PM EDT
From: "William Stallings" <ws@shore.net>
Subject: The Syntax in RFC 5322
To: Smith@Other-host.com
Cc: Jones@Yet-Another-Host.com

Hello. This section begins the actual
message body, which is delimited from the
message heading by a blank line.
```

Multipurpose Internet Mail Extensions

Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP), defined in RFC 821, or some other mail transfer protocol and RFC 5322 for electronic mail. [PARZ06] lists the following limitations of the SMTP/5322 scheme.
1. SMTP cannot transmit executable files or other binary objects. A number of schemes are in use for converting binary files into a text form that can be used by SMTP mail systems, including the popular UNIX UUencode/Uudecode scheme. However, none of these is a standard or even a de facto standard.

2. SMTP cannot transmit text data that includes national language characters, because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle nontextual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include:

• Deletion, addition, or reordering of carriage return and linefeed
• Truncating or wrapping lines longer than 76 characters
• Removal of trailing white space (tab and space characters)
• Padding of lines in a message to the same length
• Conversion of tab characters into multiple space characters

OVERVIEW:

The MIME specification includes the following elements.
1. Five new message header fields are defined, which may be included in an RFC 5322 header. These fields provide information about the body of the message.
2. A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.
3. Transfer encodings are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

In this subsection, we introduce the five message header fields. The next two subsections deal with content formats and transfer encodings.
The five header fields defined in MIME are

• **MIME-Version:** Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
• **Content-Type:** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
• **Content-Transfer-Encoding**: Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
• **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.

• **Content-Description:** A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

MIME CONTENT TYPES :

The bulk of the MIME specification is concerned with the definition of a variety of content types. This reflects the need to provide standardized ways of dealing with a wide variety of information representations in a multimedia environment.

Table lists the content types specified in RFC 2046.**There are seven different major types of content and a total of 15 subtypes.** In general, a content type declares the general type of data, and the subtype specifies a particular format for that type of data. For the text type of body, no special software is required to get the full meaning of the text aside from support of the indicated character set. The primary subtype is **plain text**, which is simply a string of ASCII characters or ISO 8859 characters. The enriched subtype allows greater formatting flexibility.

The **multipart type** indicates that the body contains multiple, independent parts. The **Content-Type** header field includes a parameter (called a boundary) that defines the delimiter between body parts. This boundary should not appear in any parts of the message. Each boundary starts on a new line and consists of two hyphens followed by the boundary value. The final boundary, which indicates the end of the last part, also has a suffix of two hyphens. Within each part, there may be an optional ordinary MIME header.

There are four subtypes of the multipart type, all of which have the same overall syntax. The **multipart/mixed subtype** is used when there are multiple independent body parts that need to be bundled in a particular order. For the multipart/parallel subtype, the order of the parts is not significant. If the recipient's system is appropriate, the multiple parts can be presented in parallel. For example, a picture or text part could be accompanied by a voice commentary that is played while the picture or text is displayed.

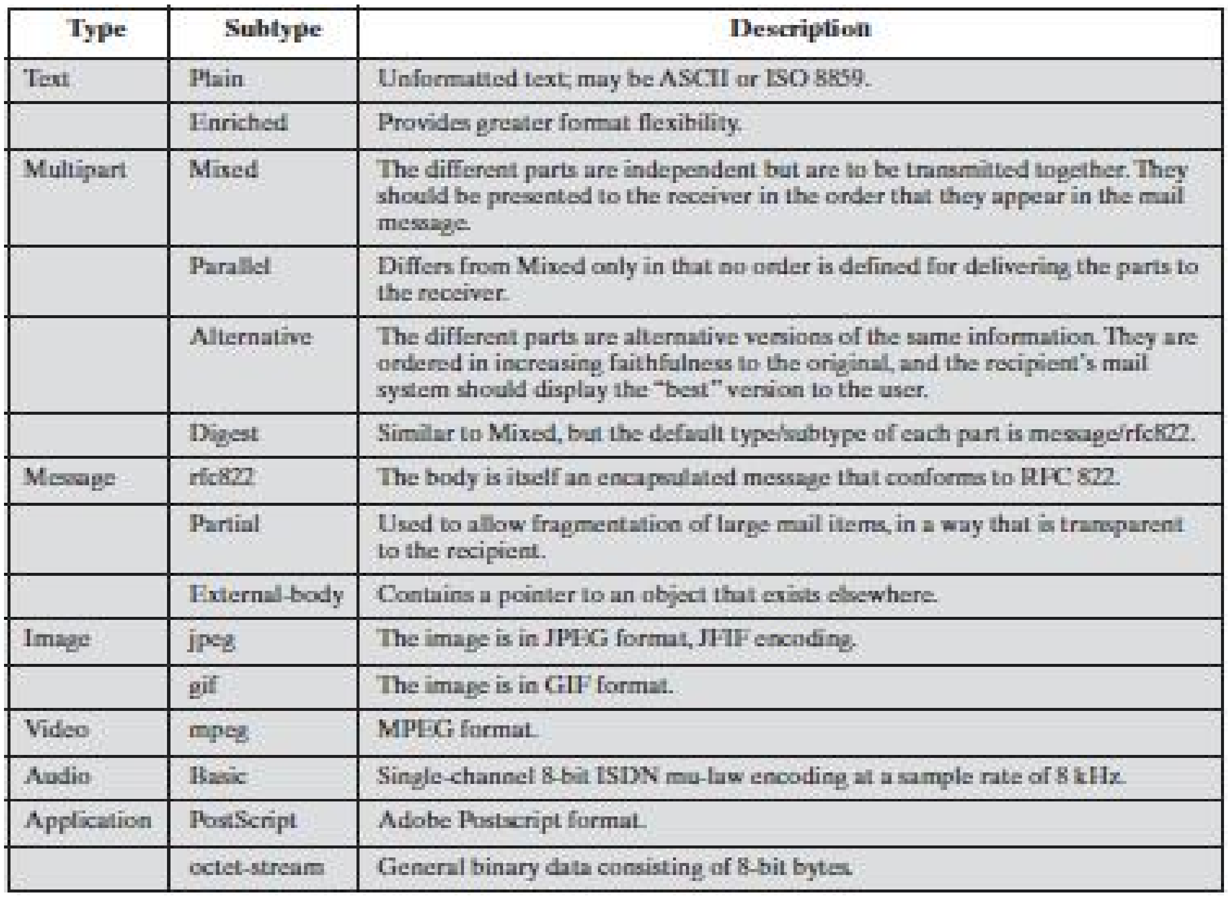
The **multipart/digest subtype** is used when each of the body parts is interpreted as an RFC 5322 message with headers. This subtype enables the construction of a message whose parts are individual messages. For example, the moderator of a group might collect e-mail messages from participants, bundle these messages, and send them out in one encapsulating MIME message.

The **message type** provides a number of important capabilities in MIME. The message/rfc822 subtype indicates that the body is an entire message, including header and body. Despite the name of this subtype, the encapsulated message may be not only a simple RFC 5322 message but also any MIME message.

The **message/partial subtype** enables fragmentation of a large message into a number of parts, which must be reassembled at the destination. For this subtype, three parameters are specified in the Content-Type: Message/Partial field: an id common to all fragments of the same message, a sequence number unique to each fragment, and the total number of fragments.

The **message/external-body subtype** indicates that the actual data to be conveyed in this message are not contained in the body. Instead, the body contains the information needed to access the data. As with the other message types, the message/ external-body subtype has an outer header and an encapsulated message with its own header. The only necessary field in the outer header is the Content-Type field, which identifies this as a message/external-body subtype. The inner header is the message header for the encapsulated message. The Content-Type field in the outer header must include an access-type parameter, which indicates the method of access, such as FTP (file transfer protocol).

The **application type** refers to other kinds of data, typically either uninterpreted binary data or information to be processed by a mail-based application.

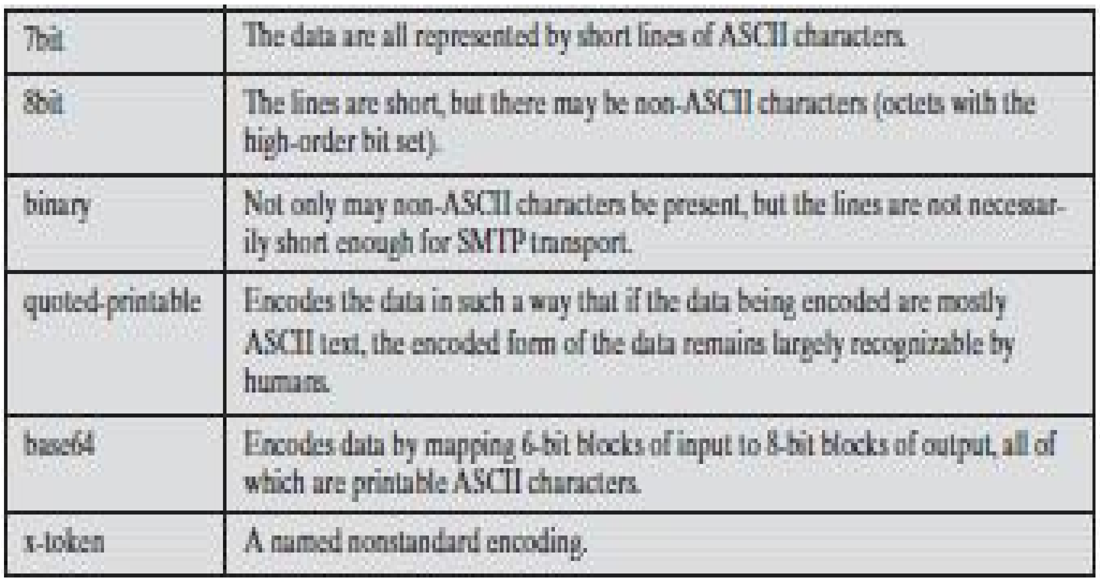


Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript format.
	octet-stream	General binary data consisting of 8-bit bytes.

MIME TRANSFER ENCODINGS:

The other major component of the MIME specification, in addition to content type specification, is a definition of transfer encodings for message bodies. The objective is to provide reliable delivery across the largest range of environments.

The MIME standard defines two methods of encoding data. The Content-Transfer-Encoding field can actually take on six values, as listed in Table However, three of these values (7bit, 8bit, and binary) indicate that no encoding has been done but provide some information about the nature of the data. For SMTP transfer, it is safe to use the 7bit form. The 8bit and binary forms may be usable in other mail transport contexts. Another Content-Transfer-Encoding value is x-token, which indicates that some other encoding scheme is used for which a name is to be supplied. This could be a vendor-specific or application-specific scheme. The two actual encoding schemes defined are quoted-printable and base64.Two schemes are defined to provide a choice between a transfer technique that is essentially human readable and one that is safe for all types of data in a way that is reasonably compact. The quoted-printable transfer encoding is useful when the data consists largely of octets that correspond to printable ASCII characters. In essence, it represents non safe characters by the hexadecimal representation of their code and introduces reversible (soft) line breaks to limit message lines to 76 characters. The base64 transfer encoding, also known as radix-64 encoding, is a common one for encoding arbitrary binary data in such a way as to be invulnerable to the processing by mail-transport programs.



7bit	The data are all represented by short lines of ASCII characters.
8bit	The lines are short, but there may be non-ASCII characters (octets with the high-order bit set).
binary	Not only may non-ASCII characters be present, but the lines are not necessarily short enough for SMTP transport.
quoted-printable	Encodes the data in such a way that if the data being encoded are mostly ASCII text, the encoded form of the data remains largely recognizable by humans.
base64	Encodes data by mapping 6-bit blocks of input to 8-bit blocks of output, all of which are printable ASCII characters.
x-token	A named nonstandard encoding.

S/MIME Functionality:

In terms of general functionality, S/MIME is very similar to PGP. Both offer the ability to sign and/or encrypt messages. In this subsection, we briefly summarize S/MIME capability functions. S/MIME provides the following functions.

• **Enveloped data:** This consists of encrypted content of any type and encrypted content encryption keys for one or more recipients.

• **Signed data:** A digital signature is formed by taking the message digest of the content to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.

• **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64.As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.

• **Signed and enveloped data:** Signed-only and encrypted-only entities may be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

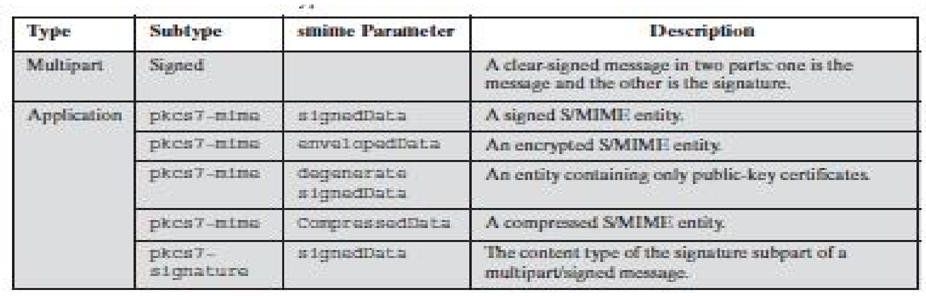
S/MIME Messages:

S/MIME makes use of a number of new MIME content types, which are shown in Table. All of the new application types use the designation PKCS. This refers to a set of public-key cryptography specifications issued by RSA Laboratories and made available for the S/MIME effort.

SECURING A MIME ENTITY: S/MIME secures a MIME entity with a signature, encryption, or both. A MIME entity may be an entire message (except for the RFC 5322 headers), or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message. The MIME entity is prepared according to the normal rules for MIME message preparation. Then the MIME entity plus some security-related data, such as algorithm identifiers and certificates, are processed by S/MIME to produce what is known as a PKCS object. A PKCS object is then treated as message content and wrapped in MIME (provided with appropriate MIME headers). This process should become clear as we look at specific objects and provide examples.

In all cases, the message to be sent is converted to canonical form. In particular, for a given type and subtype, the appropriate canonical form is used for the message content. For a multipart message, the appropriate canonical form is used for each subpart.

The use of transfer encoding requires special attention. For most cases, the result of applying the security algorithm will be to produce an object that is partially or totally represented in arbitrary binary data. This will then be wrapped in an outer MIME message, and transfer encoding can be applied at that point, typically base64. However, in the case of a multipart signed message (described in more detail later), the message content in one of the subparts is unchanged by the security process. Unless that content is 7bit, it should be transfer encoded using base64 or quoted-printable so that there is no danger of altering the content to which the signature was applied. We now look at each of the S/MIME content types.



Type	Subtype	smime Parameter	Description
Multipart	Signed		A clear-signed message in two parts: one is the message and the other is the signature.
Application	pkcs7-mime	signedData	A signed S/MIME entity.
	pkcs7-mime	envelopedData	An encrypted S/MIME entity.
	pkcs7-mime	degenerate signedData	An entity containing only public-key certificates.
	pkcs7-mime	CompressedData	A compressed S/MIME entity.
	pkcs7-signature	signedData	The content type of the signature subpart of a multipart/signed message.

4.3 IP Security

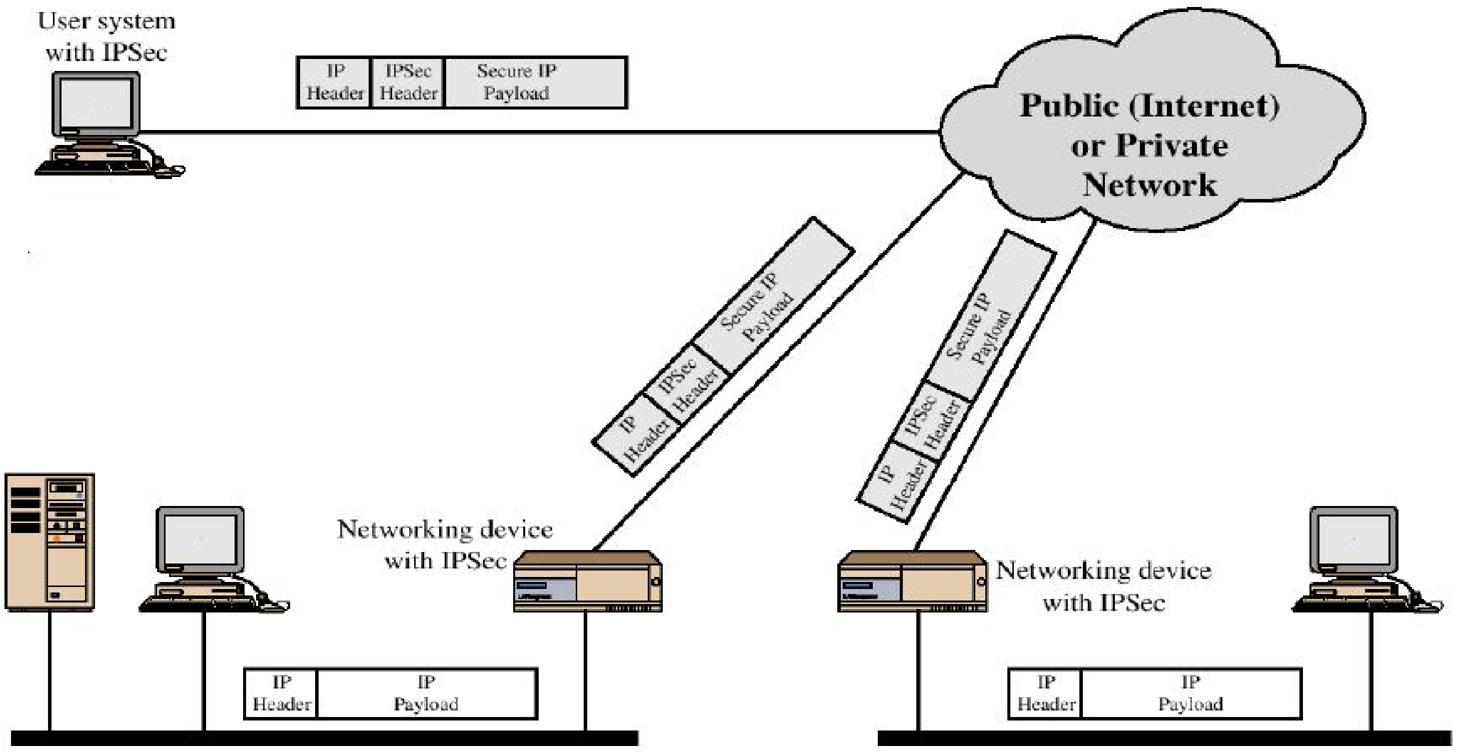
- IP security (IPsec) is a capability that can be added to either current version of the Internet Protocol (IPv4 or IPv6) by means of additional headers.
- IPsec encompasses three functional areas: authentication, confidentiality, and key management.
- IPSec is not a single protocol. Instead, IPSec provides a set of security algorithms plus a general framework that allows a pair of communicating entities to use whichever algorithms provide security appropriate for the communication.

Applications of IPSec

- Secure branch office connectivity over the Internet
- Secure remote access over the Internet
- Establishing extranet and intranet connectivity with partners

- Enhancing electronic commerce security

IP Security Scenario



The above figure is a typical scenario of IPsec usage.

Benefits of IPsec

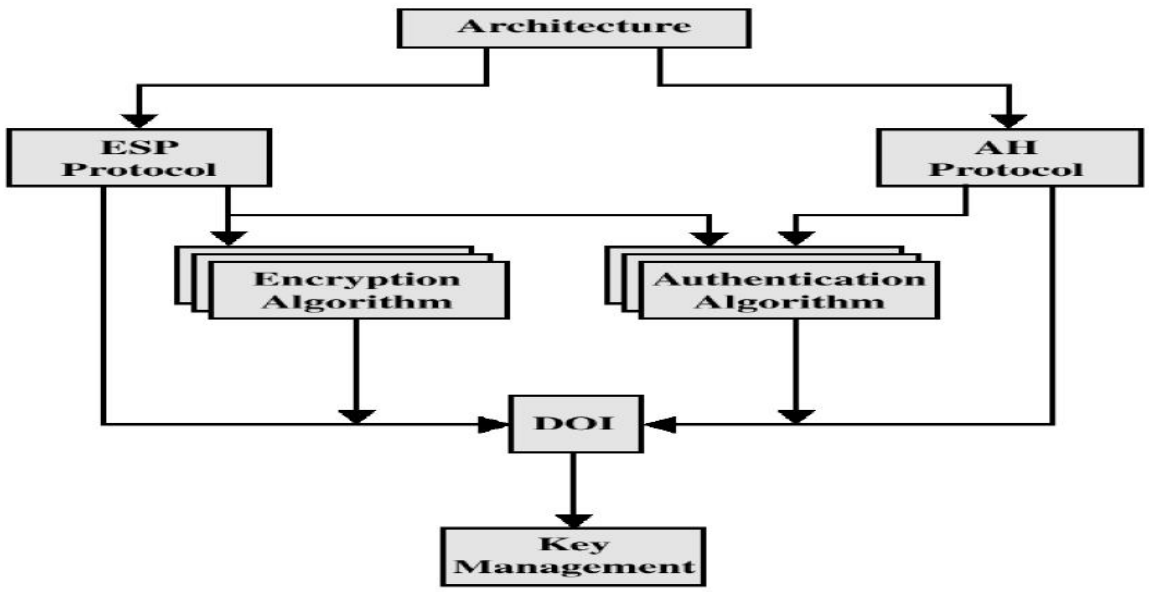
- Transparent to applications (below transport layer (TCP, UDP)
- Provide security for individual users
- IPsec can assure that:
  - ✓ A router or neighbour advertisement comes from an authorized router
  - ✓ A redirect message comes from the router to which the initial packet was sent
  - ✓ A routing update is not forged

IPsec Documents

IPsec encompasses three functional areas: authentication, confidentiality, and key management. The documents can be categorized into the following groups.

• **Architecture:** Covers the general concepts, security requirements, definitions, and mechanisms defining IPsec technology.
• **Authentication Header (AH):** AH is an extension header to provide message authentication.
• **Encapsulating Security Payload (ESP):** ESP consists of an encapsulating header and trailer used to provide encryption or combined encryption/authentication.
• **Internet Key Exchange (IKE):** This is a collection of documents describing the key management schemes for use with IPsec.
• **Cryptographic algorithms:** This category encompasses a large set of documents that define and describe cryptographic algorithms for encryption, message authentication, pseudorandom functions (PRFs), and cryptographic key exchange.
• **Other:** There are a variety of other IPsec-related RFCs, including those dealing with security policy and management information base (MIB) content.

IPSec Document Overview

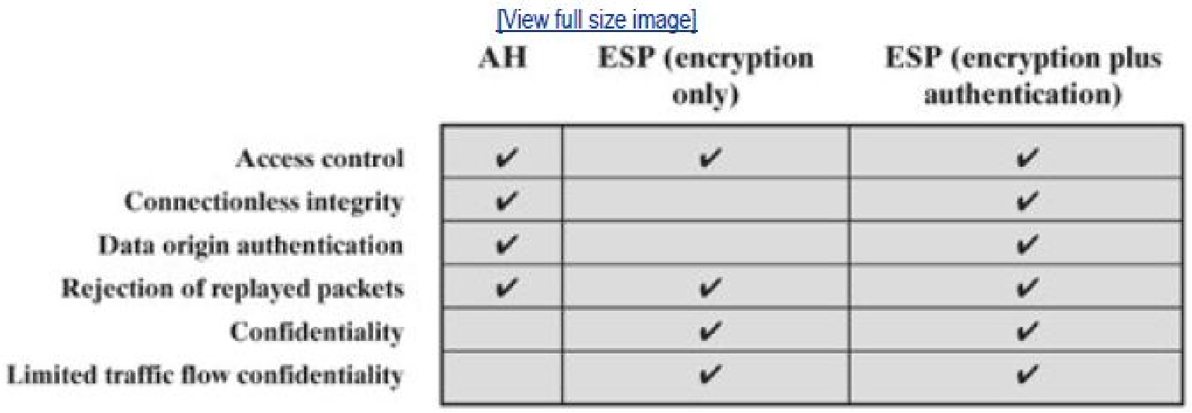


- **IPsec Services**

  IPSec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services. Two protocols are used to provide security: an authentication protocol designated by the header of the protocol, Authentication Header (AH); and a combined encryption/authentication protocol designated by the format of the packet for that protocol, Encapsulating Security Payload (ESP). The services are

- Access Control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets
- Confidentiality (encryption)
- Limited traffic flow confidentiality

Table shows which services are provided by the AH and ESP protocols. For ESP, there are two cases: with and without the authentication option. Both AH and ESP are vehicles for access control, based on the distribution of cryptographic keys and the management of traffic flows relative to these security protocols.


[View full size image]

	AH	ESP (encryption only)	ESP (encryption plus authentication)
Access control	✓	✓	✓
Connectionless integrity	✓		✓
Data origin authentication	✓		✓
Rejection of replayed packets	✓	✓	✓
Confidentiality		✓	✓
Limited traffic flow confidentiality		✓	✓

Security Associations: A key concept that appears in both the authentication and confidentiality mechanisms for IP is the security association (SA). An association is a one-way relationship between a sender and a receiver that affords security services to the traffic carried on it. If a peer relationship is needed, for two-way secure exchange, then two security associations are required. Security services are afforded to an SA for the use of AH or ESP, but not both.

A security association is uniquely identified by three parameters:

● **Security Parameters Index (SPI):** A bit string assigned to this SA and having local significance only. The SPI is carried in AH and ESP headers to enable the receiving system to select the SA under which a received packet will be processed.

● **IP Destination Address:** Currently, only unicast addresses are allowed; this is the address of the destination endpoint of the SA, which may be an end user system or a network system such as a firewall or router.

● **Security Protocol Identifier:** This indicates whether the association is an AH or ESP security association.

<u>Transport and Tunnel Modes:</u>

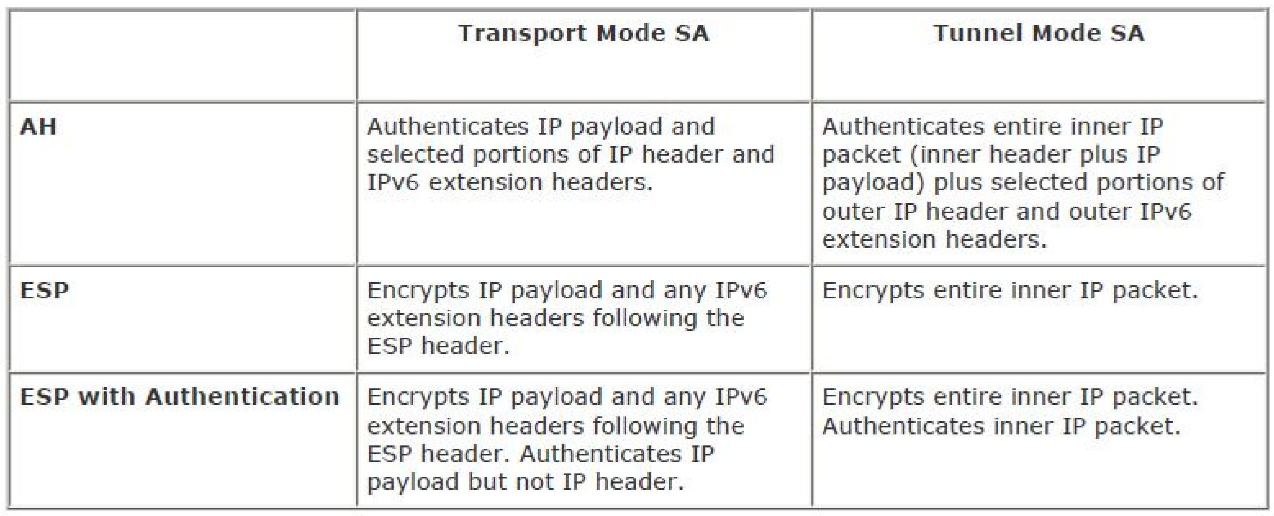
Both AH and ESP support two modes of use: transport and tunnel mode. **Transport Mode:** Transport mode provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet. Examples include a TCP or UDP segment or an ICMP packet, all of which operate directly above IP in a host protocol stack.

Typically, transport mode is used for end-to-end communication between two hosts (e.g., a client and a server, or two workstations). When a host runs AH or ESP over IPv4, the payload is the data that normally follow the IP header. For IPv6, the payload is the data that normally follow both the IP header and any IPv6 extensions headers that are present, with the possible exception of the destination options header, which may be included in the protection. ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header.

AH in transport mode authenticates the IP payload and selected portions of the IP header. Tunnel Mode Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new "outer" IP packet with a new outer IP header. The entire original, or inner, packet travels through a "tunnel" from one point of an IP network to another; no routers along the way are able to examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding to the security.

Tunnel mode is used when one or both ends of an SA are a security gateway, such as a firewall or router that implements IPSec. With tunnel mode, a number of hosts on networks behind firewalls may engage in secure communications without implementing IPSec. The unprotected packets generated by such hosts are tunneled through external networks by tunnel mode SAs set up by the IPSec software in the firewall or secure router at the boundary of the local network.

Tunnel Mode and Transport Mode Functionality



	Transport Mode SA	Tunnel Mode SA
AH	Authenticates IP payload and selected portions of IP header and IPv6 extension headers.	Authenticates entire inner IP packet (inner header plus IP payload) plus selected portions of outer IP header and outer IPv6 extension headers.
ESP	Encrypts IP payload and any IPv6 extension headers following the ESP header.	Encrypts entire inner IP packet.
ESP with Authentication	Encrypts IP payload and any IPv6 extension headers following the ESP header. Authenticates IP payload but not IP header.	Encrypts entire inner IP packet. Authenticates inner IP packet.

4.4 Authentication Header

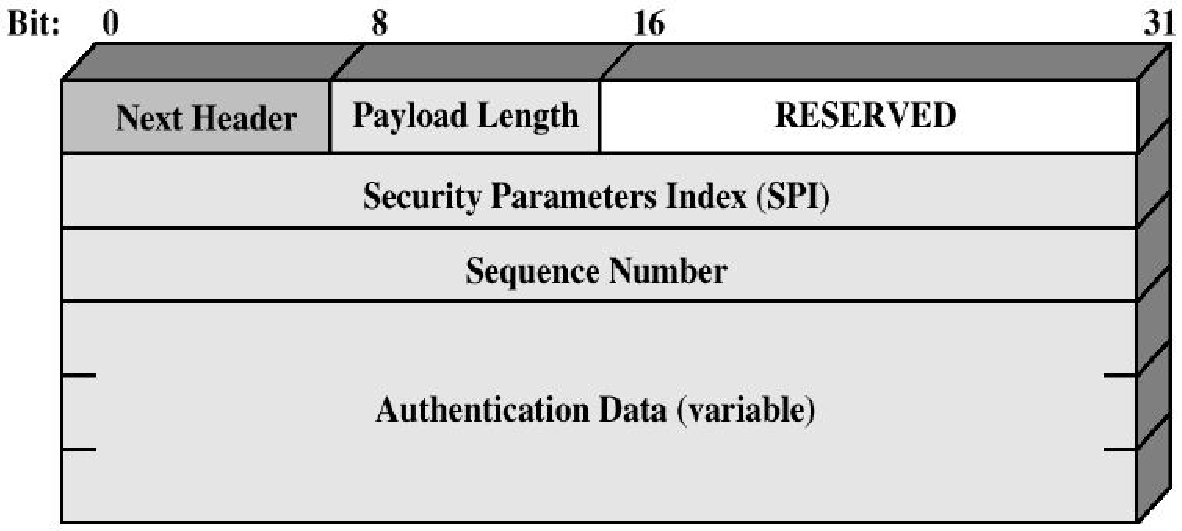
The Authentication Header provides support for data integrity and authentication of IP packets. The data integrity feature ensures that undetected modification to a packet's content in transit is not possible. The authentication feature enables an end system or network device to authenticate the user or application and filter traffic accordingly; it also prevents the address spoofing attacks observed in today's Internet.
The AH also guards against the replay attack described later in this section.
Authentication is based on the use of a message authentication code (MAC), hence the two parties must share a secret

The Authentication Header consists of the following fields :

- Next Header (8 bits): Identifies the type of header immediately following this header.
- Payload Length (8 bits): Length of Authentication Header in 32-bit words, minus 2. For example, the default length of the authentication data field is 96 bits, or three 32-bit words. With a three-word fixed header, there are a total of six words in the header, and the Payload Length field has a value of 4.
- Reserved (16 bits): For future use.
- Security Parameters Index (32 bits): Identifies a security association.

● Sequence Number (32 bits): A monotonically increasing counter value, discussed later.

● Authentication Data (variable): A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value (ICV), or MAC, for this packet, discussed later.



Bit: 0
8
16
31
Next Header
Payload Length
RESERVED
Security Parameters Index (SPI)
Sequence Number
Authentication Data (variable)

Fig: IPsec Authentication Header

Anti-Replay Service: A replay attack is one in which an attacker obtains a copy of an authenticated packet and later transmits Authentication Header it to the intended destination.

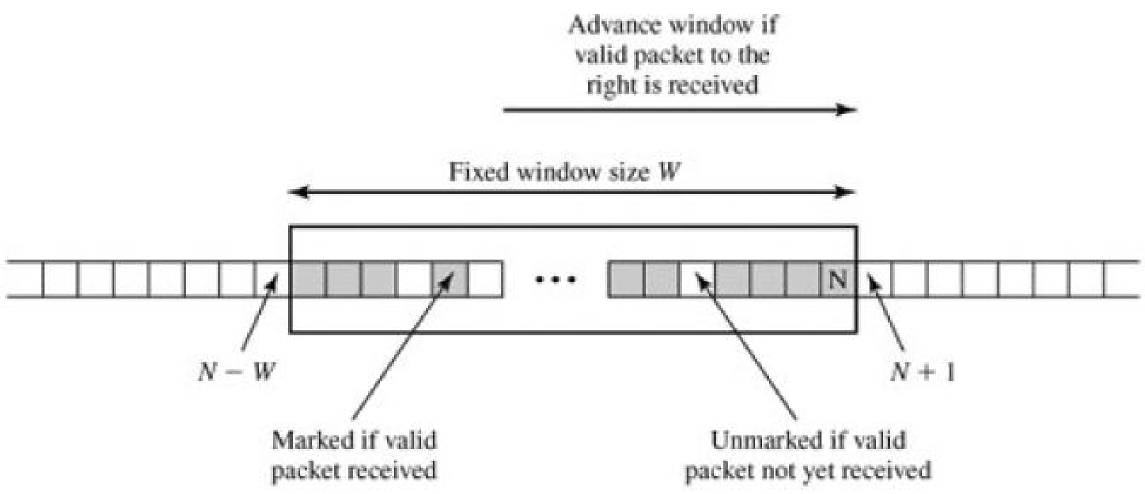
The receipt of duplicate, authenticated IP packets may disrupt service in some way or may have some other undesired consequence. The Sequence Number field is designed to thwart such attacks.

First, we discuss sequence number generation by the sender, and then we look at how it is processed by the recipient. When a new SA is established, the sender initializes a sequence number counter to 0.

Each time that a packet is sent on this SA, the sender increments the counter and places the value in the Sequence Number field. Thus, the first value to be used is 1. If anti-replay is enabled (the default), the sender must not allow the sequence number to cycle past 2^{32}-1 back to zero. Otherwise, there would be multiple valid packets with the same sequence number. If the limit of 2^{32}-1 is reached, the sender should terminate this SA and negotiate a new SA with a new key. Because IP is a connectionless, unreliable service, the protocol does not guarantee that packets will be delivered in order and does not guarantee that all packets will be delivered. Therefore, the IPSec authentication document dictates that the receiver should implement a window of size W, with a default of W = 64. The right edge of the window represents the highest sequence number, N, so far received for a valid packet. For any packet with a sequence number in the range from N W + 1 to N that has been correctly received (i.e., properly authenticated), the corresponding slot in the window is marked .Inbound processing proceeds as follows when a packet is received:

1. If the received packet falls within the window and is new, the MAC is checked. If the packet is authenticated, the corresponding slot in the window is marked.
2. If the received packet is to the right of the window and is new, the MAC is checked. If the packet is authenticated, the window is advanced so that this sequence number is the right edge of the window, and the corresponding slot in the window is marked.
3. If the received packet is to the left of the window, or if authentication fails, the packet is discarded; this is an auditable event.

Antireplay Mechanism


Advance window if valid packet to the right is received
Fixed window size W
N
N − W
N + 1
Marked if valid packet received
Unmarked if valid packet not yet received

Integrity Check Value: The Authentication Data field holds a value referred to as the Integrity Check Value. The ICV is a message authentication code or a truncated version of a code produced by a MAC algorithm. The current specification dictates that a compliant implementation must support

- HMAC-MD5-96
- HMAC-SHA-1-96

Both of these use the HMAC algorithm, the first with the MD5 hash code and the second with the SHA-1 hash code (all of these algorithms are described in Chapter 12). In both cases, the full HMAC value is calculated but then truncated by using the first 96 bits, which is the default length for the Authentication Data field.

The MAC is calculated over

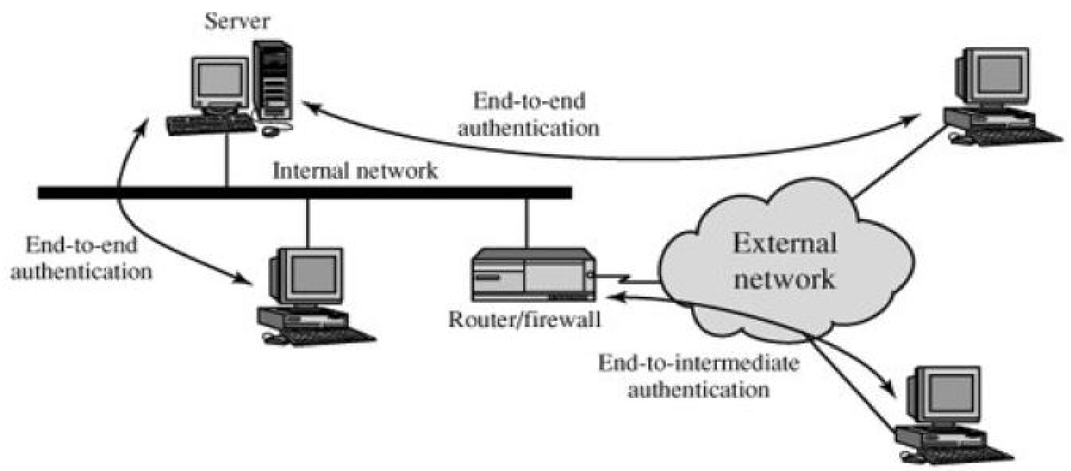
● IP header fields that either do not change in transit (immutable) or that are predictable in value upon arrival at the endpoint for the AH SA. Fields that may change in transit and whose value on arrival are unpredictable are set to zero for purposes of calculation at both source and destination.

● The AH header other than the Authentication Data field. The Authentication Data field is set to zero for purposes of calculation at both source and destination.

● The entire upper-level protocol data, which is assumed to be immutable in transit (e.g., a TCP segment or an inner IP packet in tunnel mode).

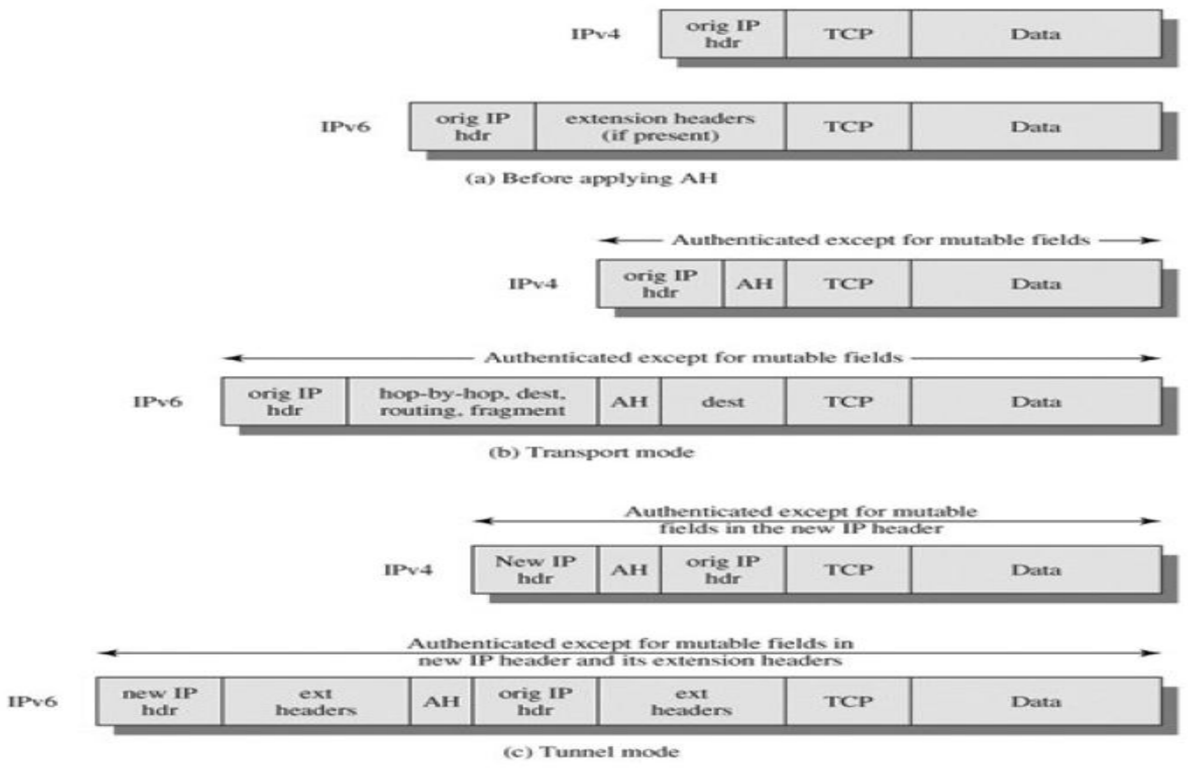
Transport and Tunnel Modes: Figure shows two ways in which the IPSec authentication service can be used. In one case, authentication is provided directly between a server and client workstations; the workstation can be either on the same network as the server or on an external network. As long as the workstation and the server share a protected secret key, the authentication process is secure. This case uses a transport mode SA. In the other case, a remote workstation authenticates itself to the corporate firewall, either for access to the entire internal network or because the requested server does not support the authentication feature. This case uses a tunnel mode SA.

End-to-End versus End-to-Intermediate Authentication


Server
End-to-end authentication
Internal network
End-to-end authentication
External network
Router/firewall
End-to-intermediate authentication

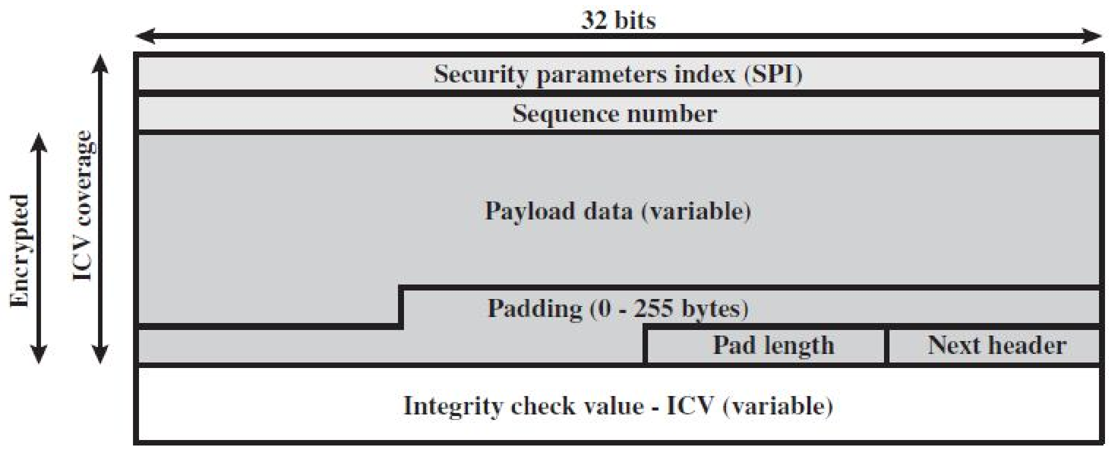
For transport mode AH using IPv4, the AH is inserted after the original IP header and before the IP payload (e.g., a TCP segment); this is shown in the upper part of Figure 16.6b. Authentication covers the entire packet, excluding mutable fields in the IPv4 header that are set to zero for MAC calculation.

Scope of AH Authentication


IPv4
orig IP hdr
TCP
Data
IPv6
orig IP hdr
extension headers (if present)
TCP
Data
(a) Before applying AH
Authenticated except for mutable fields
IPv4
orig IP hdr
AH
TCP
Data
Authenticated except for mutable fields
IPv6
orig IP hdr
hop-by-hop, dest, routing, fragment
AH
dest
TCP
Data
(b) Transport mode
Authenticated except for mutable fields in the new IP header
IPv4
New IP hdr
AH
orig IP hdr
TCP
Data
Authenticated except for mutable fields in new IP header and its extension headers
IPv6
new IP hdr
ext headers
AH
orig IP hdr
ext headers
TCP
Data
(c) Tunnel mode

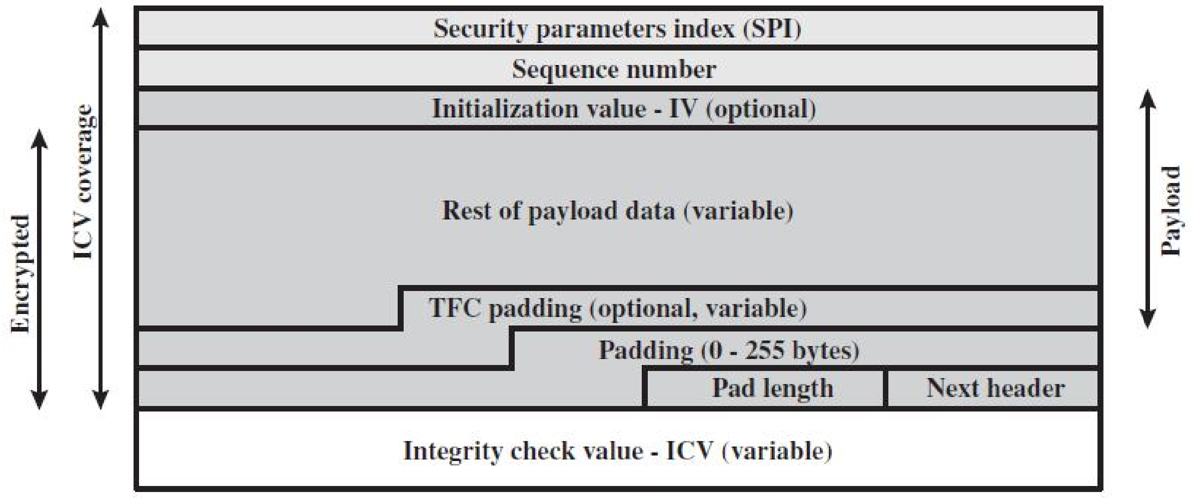
4.5 ENCAPSULATING SECURITY PAYLOAD

- ESP can be used to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service, and traffic flow confidentiality**.**
- ESP can work with a variety of encryption and authentication algorithms, including authenticated encryption algorithms the top-level format of an ESP packet.
- It contains the following fields.
  - **Security Parameters Index (32 bits):** Identifies a security association.
  - **Sequence Number (32 bits):** A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH.
  - **Payload Data (variable):** This is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption.
  - **Padding (0 – 255 bytes):** The purpose of this field is discussed later.
  - **Pad Length (8 bits):** Indicates the number of pad bytes immediately preceding this field.
  - **Next Header (8 bits):** Identifies the type of data contained in the payload data field by identifying the first header in that payload (for example, an extension header in IPv6, or an upper-layer protocol such as TCP).
  - **Integrity Check Value (variable):** A variable-length field (must be an integral number of 32-bit words) that contains the Integrity Check Value computed over the ESP packet minus the Authentication Data field



(a) Top-level format of an ESP Packet

- **An initialization value (IV), or nonce**, is present if this is required by the encryption or authenticated encryption algorithm used for ESP.
- If tunnel mode is being used, then the IPsec implementation may add traffic flow confidentiality (TFC) padding after the Payload Data and before the Padding field, as explained subsequently


Security parameters index (SPI)
Sequence number
Initialization value - IV (optional)
Rest of payload data (variable)
TFC padding (optional, variable)
Padding (0 - 255 bytes)
Pad length
Next header
Integrity check value - ICV (variable)
Encrypted
ICV coverage
Payload

(b) Substructure of payload data

Encryption and Authentication Algorithms : The Payload Data, Padding, Pad Length, and Next Header fields are encrypted by the ESP service. If the algorithm used to encrypt the payload requires cryptographic synchronization data, such as an initialization vector (IV), then these data may be carried explicitly at the beginning of the Payload Data field. If included, an IV is usually not encrypted, although it is often referred to as being part of the ciphertext. The current specification dictates that a compliant implementation must support DES in cipher block chaining (CBC) mode (described in Chapter 3). A number of other algorithms have been assigned identifiers in the DOI document and could therefore easily be used for encryption; these include

- Three-key triple DES
- RC5
- IDEA
- Three-key triple IDEA
- CAST
- Blowfish

As with AH, ESP supports the use of a MAC with a default length of 96 bits.

Padding: The Padding field serves several purposes:

● If an encryption algorithm requires the plaintext to be a multiple of some number of bytes (e.g., the multiple of a single block for a block cipher), the Padding field is used to expand the plaintext (consisting of the Payload Data, Padding, Pad Length, and Next Header fields) to the required length.

● The ESP format requires that the Pad Length and Next Header fields be right aligned within a 32- bit word. Equivalently, the ciphertext must be an integer multiple of 32 bits. The Padding field is used to assure this alignment.

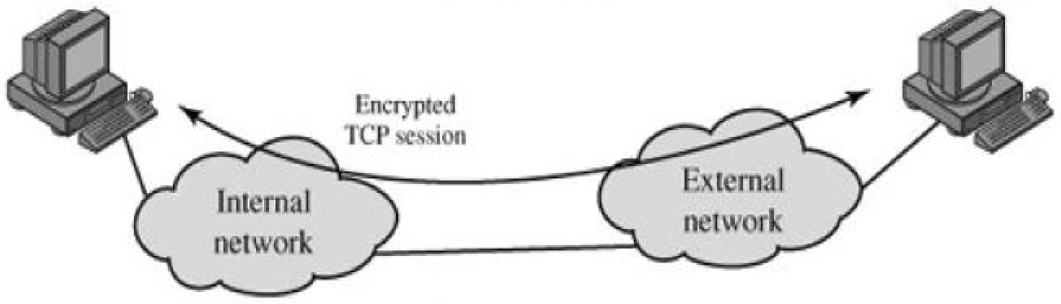
● Additional padding may be added to provide partial traffic flow confidentiality by concealing the actual length of the payload.

Transport and Tunnel Modes: Figure shows two ways in which the IPSec ESP service can be used. In the upper part of the figure, encryption (and optionally authentication) is provided directly between two hosts. Figure 16.8b shows how tunnel mode operation can be used to set up a virtual private network. In this example, an organization has four private networks interconnected across the Internet. Hosts on the internal networks use the Internet for transport of data but do not interact with other Internet-based hosts.

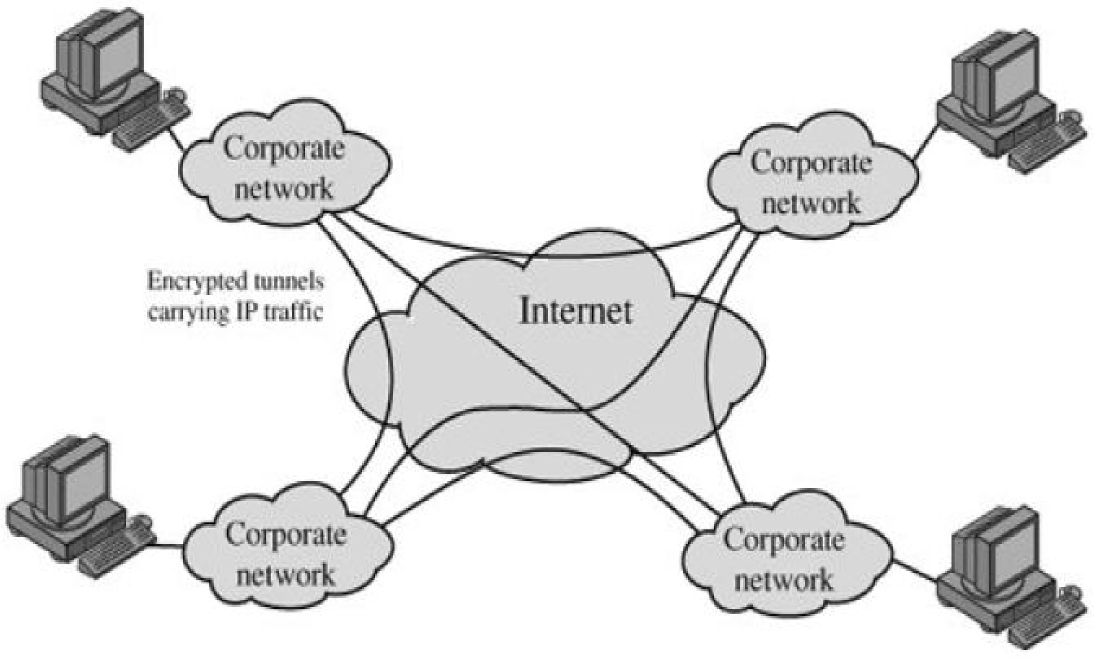
By Encapsulating Security Payload terminating the tunnels at the security gateway to each internal network, the configuration allows the hosts to avoid implementing the security capability. The former technique is support by a transport mode SA, while the latter technique uses a tunnel mode SA.

Figure: Transport-Mode vs. Tunnel-Mode Encryption

[View full size image]

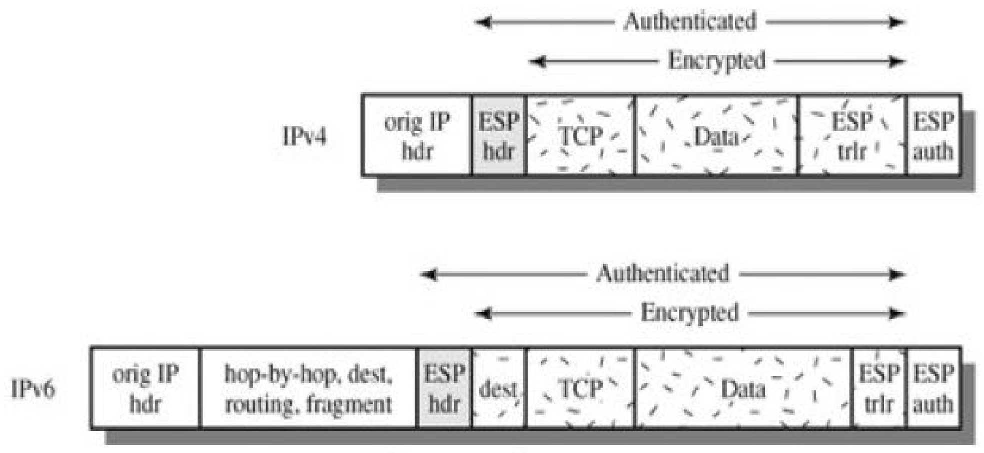


(a) Transport-level security

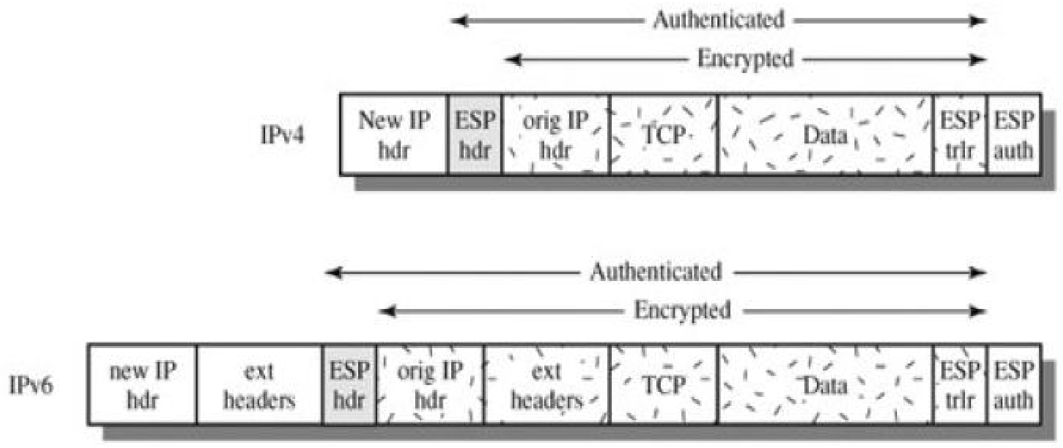


(b) A virtual private network via tunnel mode

Transport Mode ESP Transport mode ESP is used to encrypt and optionally authenticate the data carried by IP (e.g., a TCP segment), as shown in Figure. For this mode using IPv4, the ESP header is inserted into the IP packet immediately prior to the transport-layer header (e.g., TCP, UDP, ICMP) and an ESP trailer (Padding, Pad Length, and Next Header fields) is placed after the IP packet; if authentication is selected, the ESP Authentication Data field is added after the ESP trailer. The entire transport-level segment plus the ESP trailer are encrypted. Authentication covers all of the ciphertext plus the ESP header.



(a) Transport mode



(b) Tunnel mode

In the context of IPv6, ESP is viewed as an end-to-end payload; that is, it is not examined or processed by intermediate routers. Therefore, the ESP header appears after the IPv6 base header and the hop-byhop, routing, and fragment extension headers. The destination options extension header could appear before or after the ESP header, depending on the semantics desired. For IPv6, encryption covers the entire transport-level segment plus the ESP trailer plus the destination options extension header if it occurs after the ESP header. Again, authentication covers the ciphertext plus the ESP header.

Transport mode operation may be summarized as follows:

1. At the source, the block of data consisting of the ESP trailer plus the entire transport-layer segment is encrypted and the plaintext of this block

is replaced with its ciphertext to form the IP packet for transmission. Authentication is added if this option is selected.
2. The packet is then routed to the destination. Each intermediate router needs to examine and process the IP header plus any plaintext IP extension headers but does not need to examine the ciphertext.
3. The destination node examines and processes the IP header plus any plaintext IP extension headers. Then, on the basis of the SPI in the ESP header, the destination node decrypts the remainder of the packet to recover the plaintext transport-layer segment.
Transport mode operation provides confidentiality for any application that uses it, thus avoiding the need to implement confidentiality in every individual application. This mode of operation is also reasonably efficient, adding little to the total length of the IP packet. One drawback to this mode is that it is possible to do traffic analysis on the transmitted packets.
Tunnel Mode ESP: Tunnel mode ESP is used to encrypt an entire IP packet.For this mode, the ESP header is prefixed to the packet and then the packet plus the ESP trailer is encrypted. This method can be used to counter traffic analysis.
Because the IP header contains the destination address and possibly source routing directives and hop-by-hop option information, it is not possible simply to transmit the encrypted IP packet prefixed by the ESP header. Intermediate routers would be unable to process such a packet. Therefore, it is necessary to encapsulate the entire block (ESP header plus ciphertext plus Authentication Data, if present) with a new IP header that will contain sufficient information for routing but not for traffic analysis. Whereas the transport mode is suitable for protecting connections between hosts that support the ESP feature, the tunnel mode is useful in a configuration that includes a firewall or other sort of security gateway that protects a trusted network from external networks. In this latter case, encryption occurs only between an external host and the security gateway or between two security gateways. This relieves hosts on the internal network of the processing burden of encryption and simplifies the key distribution task by reducing the number of needed keys. Further, it thwarts traffic analysis based on ultimate destination.

UNIT-IV
Assignment-Cum-Tutorial Questions
SECTION-A

Objective Questions

1. In PGP services, digital signatures uses ________ algorithms. [ ]

a) DSS/RSA b) RSA/SHA c) either (a) or (b) d) DES

2. ________ operates in the transport mode or the tunnel mode. [ ]

a) IPSec b) PGP c) SMIME d) DES

3. IPSec defines two protocols: ______ and _______. [ ]

a)AH,SMIME b) PGP,ESP c) AH,ESP d) AH,PGP

5. _____ provides authentication at the IP level. [ ]

a) AH b) ESP c) PGP d) SMIME

6. To provide transparency for e-mail applications, an encrypted message converted to ASCII string is the function of ____________ [ ]

a) digital signature b) e-mail compatibility

c) segmentation d) compression

7. Combination of SHA-1 and RSA provides an effective __________[ ]

a) e-mail application b) digital signature

c) confidentiality d) authentication

8. Multiple/alternative subtype is used when______ [ ]

a) multiple independent body parts that need to be bundled

b) multiple parts can be present in parallel

c) different representations of same information

d)fragmentation of large messages into number of parts

9. ___provides privacy, integrity, and authentication in e-mail. [ ]
a) IPSec b) SSL c) PGP (d) S/MIME

10. In PGP, to exchange e-mail messages, a user needs a ring of ______ keys. [ ]

a) secret b) public c) either (a) or (b) d) both (a) and (b)

11. IP security uses _________ routing protocol. [ ]

a) OPSF b) TCP c) OSPF d) UDP

12. Anti-replay mechanism uses the window size of _________ [ ]

a) w-1 b) w+1 c) 2w d) w

13. IPSec in the _____ mode does not protect the IP header. [ ]

a) transport b) tunnel

c) either (a) or (b) d) neither (a) nor (b)

14. A ______ layer security protocol provides end-to-end security services for applications. [ ]

a) datalink b)network c)transport d) application

15. Nonce is a ________ [ ]

a) generated random number

b) locally generated pseudorandom number

c) globally generated pseudorandom number

d) both b) and c)

SECTION-B

SUBJECTIVE QUESTIONS

1. Explain PGP cryptographic functions

2. With neat diagram, explain PGP message transmission and reception

3. Explain how email messages are protected using S/MIME signing and encryption.

4. Explain MIME encoding techniques.

5. Describe S/MIME functions and certificate processing.

6. Explain enhanced security services of S/MIME.

7. Explain the scope of ESP encryption and authentication in tunnel mode.

8. Write short notes on AH.

9. What is Radix 64 format? What is its use in PGP?

10. List different MIME content types.

11. List different encryption and authentication algorithms which are used for AH and ESP.

12. Briefly explain replay attack.

13. Write some of the applications of IPSec.

14. Differentiate the packet structure of ESP and AH

INFORMATION SECURITY

UNIT V
Learning Material

Unit – V

Objectives:

Present an overview of techniques for remote user authentication, Kerberos, Summarize Web Security threats and Web traffic security approaches, overview of SSL & TLS.

Syllabus: Transport-Le Security

Web Security Requirements, Secure Socket Layer (SSL) and Transport Layer Security (TLS), Secure Electronic Transaction (SET).

Outcomes:

Students will be able to

- Proposing new strategies to secure threats on web.
- To Analyze and compare various security levels at transport layer.

WEB SECURITY REQUIREMENTS

The World Wide Web is fundamentally a client/server application running over the Internet and TCP/IP intranets. Web presents new challenges not generally appreciated in the context of computer and network security.

• The Internet is two-way. Unlike traditional publishing environments—even electronic publishing systems involving teletext, voice response, or fax-back— the Web is vulnerable to attacks on the Web servers over the Internet.

• The Web is increasingly serving as a highly visible outlet for corporate and product information and as the platform for business transactions. Reputations can be damaged and money can be lost if the Web servers are subverted.

• Although Web browsers are very easy to use, Web servers are relatively easy to configure and manage, and Web content is increasingly easy to develop, the underlying software is extraordinarily complex. This complex software may hide many potential security flaws. The short history of the Web is filled with examples of new and upgraded systems, properly installed, that are vulnerable
to a variety of security attacks.

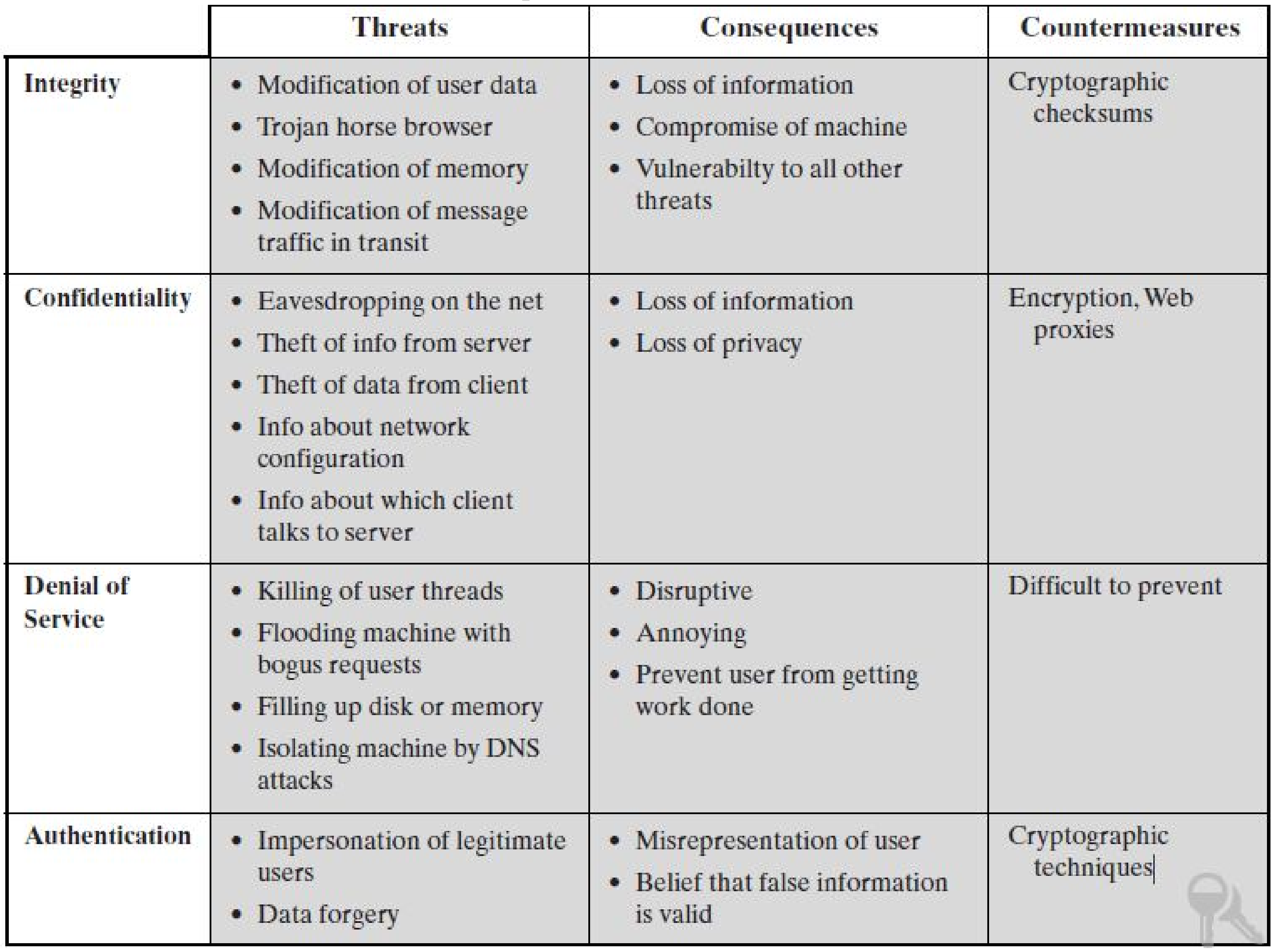
• A Web server can be exploited as a launching pad into the corporation'sor agency's entire computer complex. Once the Web server is subverted, an attacker may be able to gain access to data and systems not part of the Web itself but connected to the server at the local site.

• Casual and untrained (in security matters) users are common clients for Web-based services. Such users are not necessarily aware of the security risks that exist and do not have the tools or knowledge to take effective countermeasures.

Web Traffic Security Approaches

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

Table: A Comparison of Threats on the Web



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

SECURE SOCKET LAYER (SSL)

Secure Socket Layer (SSL) provides security services between TCP and applications that use TCP.

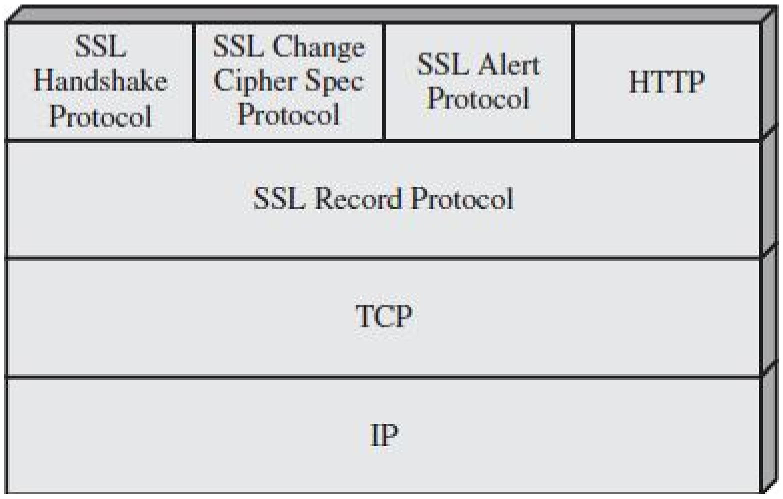
SSL Architecture

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

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

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

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



SSL Protocol Stack

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

A session state is defined by the following parameters.

• **Session identifier:** An arbitrary byte sequence chosen by the server to identify an active or resumable session state.

• **Peer certificate:** An X509.v3 certificate of the peer. This element of the state may be null.

• **Compression method:** The algorithm used to compress data prior to encryption.

• **Cipher spec:** Specifies the bulk data encryption algorithm (such as null,AES, etc.) and a hash algorithm (such as MD5 or SHA-1) used for MAC calculation. It also defines cryptographic attributes such as the hash size.

• **Master secret:** 48-byte secret shared between the client and server.

• **Is resumable:** A flag indicating whether the session can be used to initiate new connections.

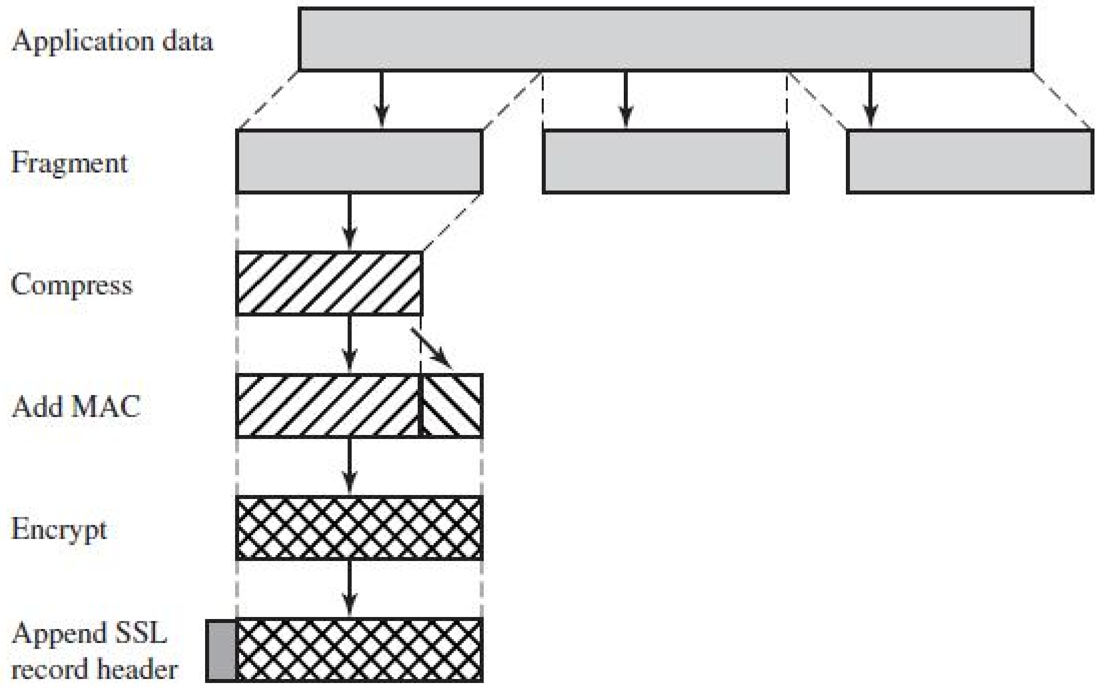
A connection state is defined by the following parameters.

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

SSL Record Protocol

The SSL Record Protocol provides two services for SSL connections:
• **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.
• **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).
The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users.

The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 2^{14} bytes (16384 bytes) or less. Next, **compression** is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes. In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm
is null.
The next step in processing is to compute a **message authentication code** over the compressed data. For this purpose, a shared secret key is used. The calculation is defined as



SSL Record Protocol Operation

hash(MAC_write_secret || pad_2||hash(MAC_write_secret || pad_1||seq_num||SSLCompressed.type || SSLCompressed.length ||SSLCompressed.fragment))

where

|| = concatenation

MAC_write_secret = shared secret key

hash = cryptographic hash algorithm; either MD5 or SHA-1

pad_1 = the byte 0x36 (0011 0110) repeated 48 times (384 bits) for MD5 and 40 times (320 bits) for SHA-1

pad_2 = the byte 0x5C (0101 1100) repeated 48 times for MD5 and 40 times for SHA-1

seq_num = the sequence number for this message

SSLCompressed.type = the higher-level protocol used to process this fragment

SSLCompressed.length = the length of the compressed fragment

SSLCompressed.fragment = the compressed fragment (if compression is not used, this is the plaintext fragment).

Note that this is very similar to the HMAC algorithm. The difference is that the two pads are concatenated in SSLv3 and are XORed in HMAC. The SSLv3 MAC algorithm is based on the original Internet draft for HMAC, which used concatenation.
The final version of HMAC (defined in RFC 2104) uses the XOR.
Next, the compressed message plus the MAC are encrypted using symmetric encryption. Encryption may not increase the content length by more than 1024 bytes, so that the total length may not exceed $2^{14} + 2048$.

The final step of SSL Record Protocol processing is to prepare a header consisting of the following fields:
• **Content Type (8 bits):** The higher-layer protocol used to process the enclosed fragment.
• **Major Version (8 bits):** Indicates major version of SSL in use. For SSLv3, the value is 3.
• **Minor Version (8 bits):** Indicates minor version in use. For SSLv3, the value is 0.
• **Compressed Length (16 bits):** The length in bytes of the plaintext fragment (or compressed fragment if compression is used).
The content types that have been defined are change_cipher_spec,alert, handshake, and application_data.

TRANSPORT LAYER SECURITY (TLS)

TLS is an IETF standardization initiative whose goal is to produce an Internet standard version of SSL. TLS is defined as a Proposed Internet Standard in RFC 5246. RFC 5246 is very similar to SSLv3.

Version Number

The TLS Record Format is the same as that of the SSL Record Format and the fields in the header have the same meanings. The one difference is in version values.

Message Authentication Code

There are two differences between the SSLv3 and TLS MAC schemes: the actual algorithm and the scope of the MAC calculation. TLS makes use of the HMAC algorithm defined in RFC 2104. HMAC is defined as

$$HMACK(M)= H[(K+\oplus opad)||H[(K+\oplus ipad)||M]]$$

where
H = embedded hash function (for TLS, either MD5 or SHA-1)
M = message input to HMAC
K+ = secret key padded with zeros on the left so that the result is equal
to the block length of the hash code (for MD5 and SHA-1, block

length = 512 bits)
ipad = 00110110 (36 in hexadecimal) repeated 64 times (512 bits)
opad = 01011100 (5C in hexadecimal) repeated 64 times (512 bits)

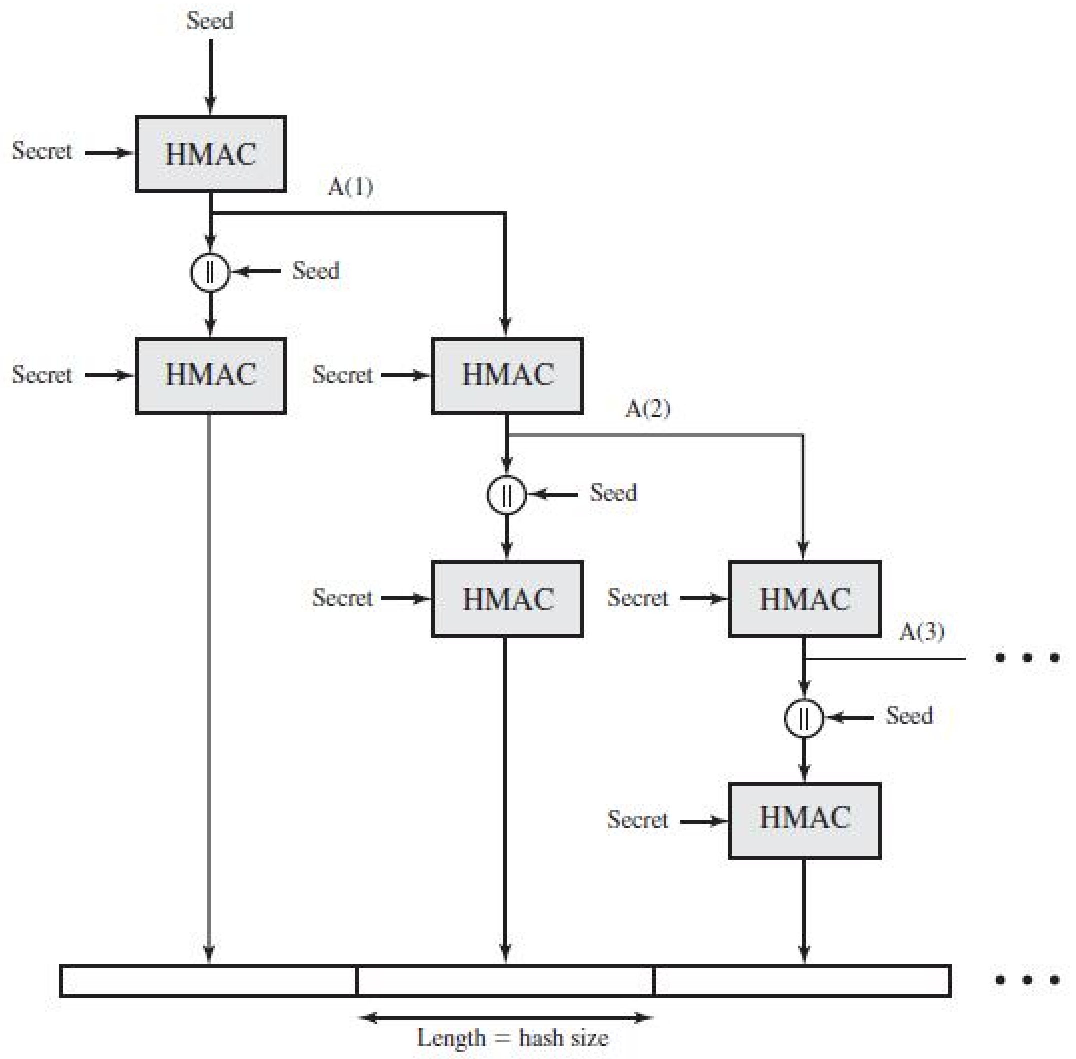
SSLv3 uses the same algorithm, except that the padding bytes are concatenated with the secret key rather than being XORed with the secret key padded to the block length. The level of security should be about the same in both cases.
For TLS, the MAC calculation encompasses the fields indicated in the following expression:

```
MAC(MAC_write_secret,seq_num || TLSCompressed.type ||
          TLSCompressed.version || TLSCompressed.length ||
                    TLSCompressed.fragment)
```

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

Pseudorandom Function
TLS makes use of a pseudorandom function referred to as PRF to expand secrets into blocks of data for purposes of key generation or validation. The objective is to make use of a relatively small shared secret value but to generate longer blocks of data in a way that is secure from the kinds of attacks made on hash functions and MACs.The PRF is based on the data expansion function given as

```
P_hash(secret, seed)= HMAC_hash(secret,A(1) || seed) ||
                      HMAC_hash(secret, A(2) || seed) ||
                      HMAC_hash(secret, A(3) || seed) || . . .
          where A() is defined as
          A(0) = seed
          A(i) = HMAC_hash(secret,A(i – 1))
```



TLS Function P_hash(secret, seed)

The data expansion function makes use of the HMAC algorithm with either MD5 or SHA-1 as the underlying hash function. As can be seen, P_hash can be iterated as many times as necessary to produce the required quantity of data. For example, if P_SHA-1 was used to generate 64 bytes of data, it would have to be iterated four times, producing 80 bytes of data of which the last 16 would be discarded. In this case, P_MD5 would also have to be iterated four times, producing exactly 64 bytes of data. Note that each iteration involves two executions of HMAC—each of which in turn involves two executions of the underlying hash algorithm.

To make PRF as secure as possible, it uses two hash algorithms in a way that should guarantee its security if either algorithm remains secure. PRF is defined as

PRF(secret, label, seed) = P_hash(S1,label || seed)

PRF takes as input a secret value, an identifying label, and a seed value and produces an output of arbitrary length.

Alert Codes

TLS supports all of the alert codes defined in SSLv3 with the exception of no_certificate. A number of additional codes are defined in TLS; of these, the following are always fatal.

• **record_overflow:** A TLS record was received with a payload (ciphertext) , whose length exceeds bytes, or the ciphertext decrypted to a length of greater than bytes.

• **unknown_CA:** A valid certificate chain or partial chain was received, but the certificate was not accepted because the CA certificate could not be located or could not be matched with a known, trusted CA.

• **access_denied:** A valid certificate was received, but when access control was applied, the sender decided not to proceed with the negotiation.

• **decode_error:** A message could not be decoded, because either a field was out of its specified range or the length of the message was incorrect.

• **protocol_version:** The protocol version the client attempted to negotiate is recognized but not supported.

• **insufficient_security:** Returned instead of handshake_failure when a negotiation has failed specifically because the server requires ciphers more secure than those supported by the client.

• **unsupported_extension:** Sent by clients that receive an extended server hello containing an extension not in the corresponding client hello.

• **internal_error:** An internal error unrelated to the peer or the correctness of the protocol makes it impossible to continue.

• **decrypt_error:** A handshake cryptographic operation failed, including being unable to verify a signature, decrypt a key exchange, or validate a finished message.

The remaining alerts include the following.

• **user_canceled:** This handshake is being canceled for some reason unrelated to a protocol failure.

• **no_renegotiation:** Sent by a client in response to a hello request or by the server in response to a client hello after initial handshaking. Either of these messages would normally result in renegotiation, but this alert indicates that the sender is not able to renegotiate.This message is always a warning.

SECURE ELECTRONICS TRANSACTIONS (SET)

SET is an open encryption and security specification designed to protect credit card transactions on the Internet. The current version, SETv1, emerged from a call for security standards by MasterCard and Visa in February 1996. A wide range of companies were involved in developing the initial specification, including IBM, Microsoft, Netscape, RSA, Terisa, and Verisign. Beginning in 1996, there have been numerous tests of the concept, and by 1998 the first wave of SET-compliant products was available.

SET is not itself a payment system. Rather it is a set of security protocols and formats that enables users to employ the existing credit card payment infrastructure on an open network, such as the Internet, in a secure fashion. In essence, SET provides three services:

- Provides a secure communications channel among all parties involved in a transaction
- Provides trust by the use of X.509v3 digital certificates
- Ensures privacy because the information is only available to parties in a transaction when and where necessary

Requirements

The SET specification lists the following business requirements for secure payment processing with credit cards over the Internet and other networks:

- **Provide confidentiality of payment and ordering information:** It is necessary to assure cardholders that this information is safe and accessible only to the intended recipient. Confidentiality also reduces the risk of fraud by either party to the transaction or by malicious third parties. SET uses encryption to provide confidentiality.
- **Ensure the integrity of all transmitted data:** That is, ensure that no changes in content occur during transmission of SET messages. Digital signatures are used to provide integrity.
- **Provide authentication that a cardholder is a legitimate user of a credit card account:** A mechanism that links a cardholder to a specific account number reduces the incidence of fraud and the overall cost of payment processing. Digital signatures and certificates are used to verify that a cardholder is a legitimate user of a valid account.
- **Provide authentication that a merchant can accept credit card transactions through its relationship with a financial institution:** This is the complement to the preceding requirement. Cardholders need to be able to identify merchants with whom they can conduct secure transactions. Again, digital signatures and certificates are used.
- **Ensure the use of the best security practices and system design techniques to protect all legitimate parties in an electronic commerce transaction:** SET is a well-tested specification based on highly secure cryptographic algorithms andprotocols.
- **Create a protocol that neither depends on transport security mechanisms nor prevents their use:** SET can securely operate over a "raw" TCP/IP stack. However, SET does not interfere with the use of other security mechanisms, such as IPSec and SSL/TLS.

- **Facilitate and encourage interoperability among software and network providers:** The SET protocols and formats are independent of hardware platform, operating system, and Web software.

Key Features of SET

To meet the requirements just outlined, SET incorporates the following features:

- **Confidentiality of information:** Cardholder account and payment information is secured as it travels across the network. An interesting and important feature of SET is that it prevents the merchant from learning the cardholder's credit card number; this is only provided to the issuing bank. Conventional encryption by DES is used to provide confidentiality.
- **Integrity of data:** Payment information sent from cardholders to merchants includes order information, personal data, and payment instructions. SET guarantees that these message contents are not altered in transit. RSA digital signatures, using SHA-1 hash codes, provide message integrity. Certain messages are also protected by HMAC using SHA-1.
- **Cardholder account authentication:** SET enables merchants to verify that a cardholder is a legitimate user of a valid card account number. SET uses X.509v3 digital certificates with RSA signatures for this purpose.
- **Merchant authentication:** SET enables cardholders to verify that a merchant has a relationship with a financial institution allowing it to accept payment cards. SET uses X.509v3 digital certificates with RSA signatures for this purpose.

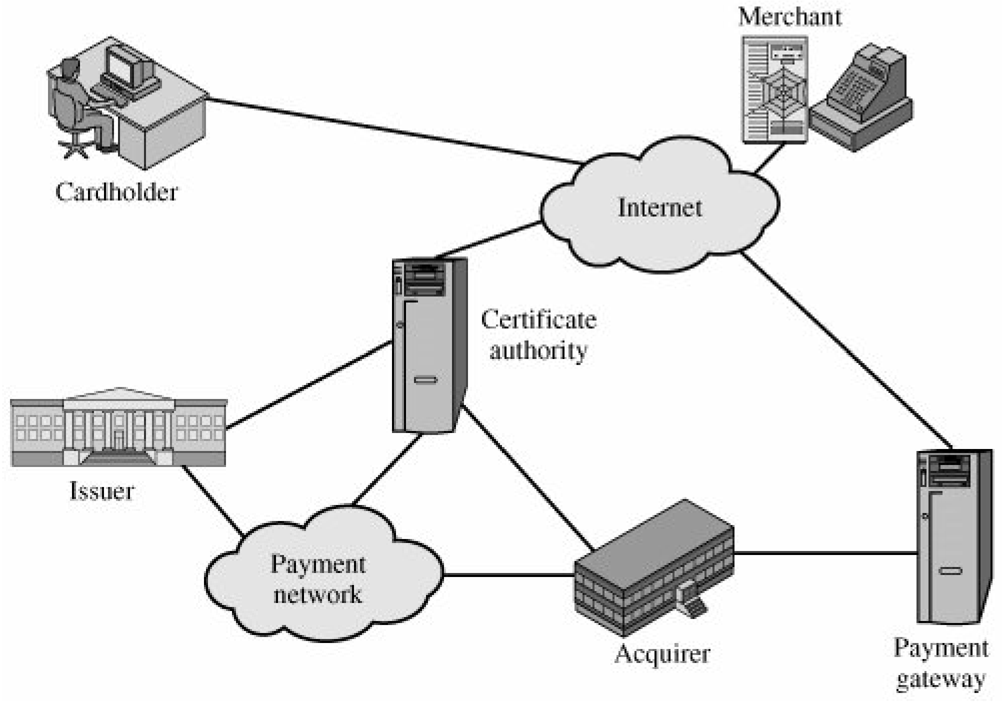
SET Participants

The participants in the SET system, which include the following:

- **Cardholder:** In the electronic environment, consumers and corporate purchasers interact with merchants from personal computers over the Internet. A cardholder is an authorized holder of a payment card (e.g., MasterCard, Visa) that has been issued by an issuer.
- **Merchant:** A merchant is a person or organization that has goods or services to sell to the cardholder. Typically, these goods and services are offered via a Web site or by electronic mail. A merchant that accepts payment cards must have a relationship with an acquirer.
- **Issuer**: This is a financial institution, such as a bank, that provides the cardholder with the payment card. Typically, accounts are applied for and opened by mail or in person. Ultimately, it is the issuer that is responsible for the payment of the debt of the cardholder.
- **Acquirer**: This is a financial institution that establishes an account with a merchant and processes payment card authorizations and payments. Merchants will usually accept more

than one credit card brand but do not want to deal withmultiple bankcard associations or with multiple individual issuers. The acquirer provides authorization to the merchant that a given card account is active and that the proposed purchase does not exceed the credit limit. The acquirer also provides electronic transfer of payments to the merchant's account. ubsequently, the acquirer is reimbursed by the issuer over some sort of payment network for electronic funds transfer.

- **Payment gateway**: This is a function operated by the acquirer or a designated third party that processes merchant payment messages. The payment gateway interfaces between SET and the existing bankcard payment networks for authorization and payment functions. The merchant exchanges SET messages with the payment gateway over the Internet, while the payment gateway has some direct or network connection to the acquirer's financial processing system.
- **Certification authority (CA):** This is an entity that is trusted to issue X.509v3 public-key certificates for cardholders, merchants, and payment gateways. The success of SET will depend on the existence of a CA infrastructure available for this purpose. As was discussed in previous chapters, a hierarchy of CAs is used, so that participants need not be directly certified by a root authority.



Secure Electronic Commerce Components

The sequence of events that are required for a transaction. We will then look at some of the cryptographic details.

1. **The customer opens an account.** The customer obtains a credit card account, such as MasterCard or Visa, with a bank that supports electronic payment and SET.

2. **The customer receives a certificate.** After suitable verification of identity, the customer receives an X.509v3 digital certificate, which is signed by the bank. The certificate verifies the customer's RSA public key and its expiration date. It also establishes a relationship, guaranteed by the bank, between the customer's key pair and his or her credit card.
3. **Merchants have their own certificates.** A merchant who accepts a certain brand of card must be in possession of two certificates for two public keys owned by the merchant: one for signing messages, and one for key exchange. The merchant also needs a copy of the payment gateway's public-key certificate.
4. **The customer places an order.** This is a process that may involve the customer first browsing through the merchant's Website to select items and determine the price. The customer then sends a list of the items to be purchased to the merchant, whoreturns an order form containing the list of items, their price, a total price, and an order number.
5. **The merchant is verified.** In addition to the order form, the merchant sends a copy of its certificate, so that the customer can verify that he or she is dealing with a valid store.
6. **The order and payment are sent.** The customer sends both order and payment information to the merchant, along with the customer's certificate. The order confirms the purchase of the items in the order form. The payment contains credit card details. The payment information is encrypted in such a way that it cannot be read by the merchant. The customer's certificate enables the merchant to verify the customer.
7. **The merchant requests payment authorization.** The merchant sends the payment information to the payment gateway, requesting authorization that the customer's available credit is sufficient for this purchase.
8. **The merchant confirms the order.** The merchant sends confirmation of the order to the customer.
9. **The merchant provides the goods or service.** The merchant ships the goods or provides the service to the customer.
10. **The merchant requests payment.** This request is sent to the payment gateway, which handles all of the payment processing.

Dual Signature

Before looking at the details of the SET protocol, let us discuss an important innovation introduced in SET: the dual signature. The purpose of the dual signature is to link two messages that are intended for two different recipients. In this case, the customer wants to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to know the customer's credit card number, and the bank does not need to know the details of the customer's order. The customer is afforded extra protection in terms of privacy by keeping these two items separate. However, the two items must be linked in a way that can be used to resolve disputes if necessary. The link is needed so that the customer can prove that this payment is intended for this order and not for some other goods or service.

The customer takes the hash (using SHA-1) of the PI and the hash of the OI. These two hashes are then concatenated and the hash of the result is taken. Finally, the customer encrypts the final hash with his or her private signature key, creating the dual signature. The operation can be summarized as

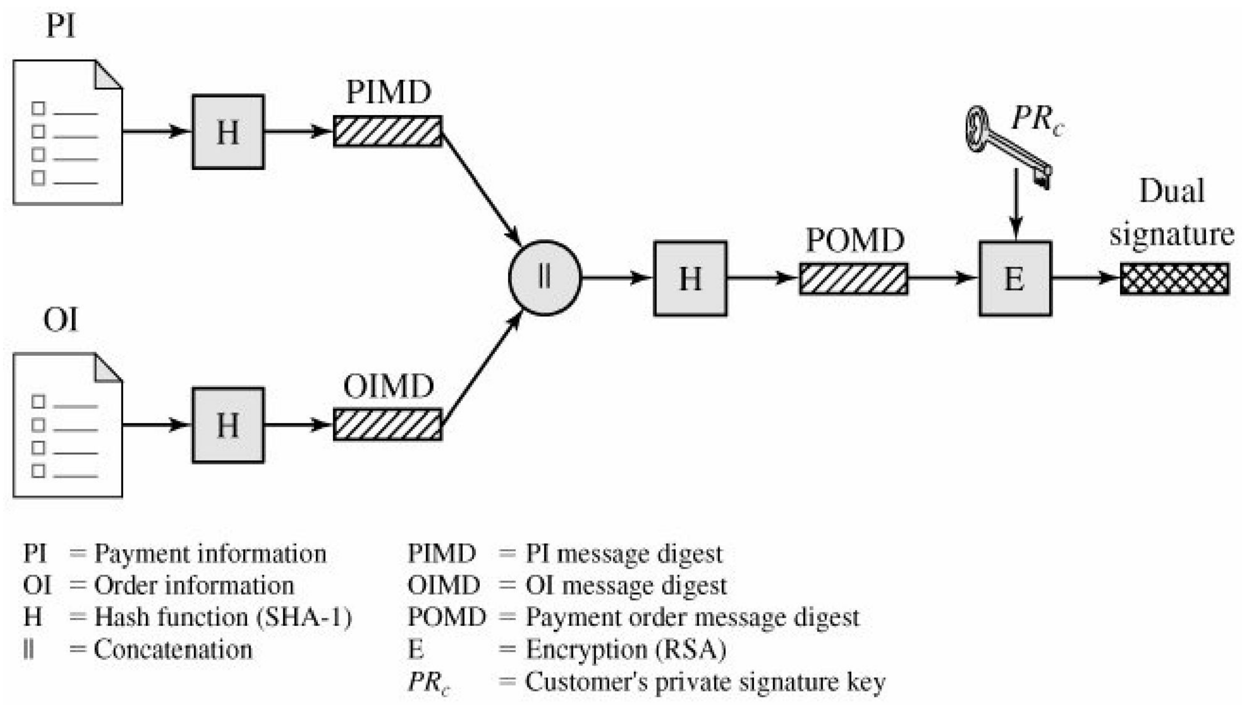
$$\mathbf{DS = E(\mathit{PRc}, [H(H(PI)||H(OI)])}$$

where *PRc* is the customer's private signature key. Now suppose that the merchant is in possession of the dual signature (DS), the OI, and the message digest for the PI (PIMD). The merchant also has the public key of the customer, taken from the customer's certificate. Then the merchant can compute the quantities

$$\mathbf{H(PIMS||H[OI]);\ D(\mathit{PUc}, DS)}$$

where *PUc* is the customer's public signature key. If these two quantities are equal, then the merchant has verified the signature. Similarly, if the bank is in possession of DS, PI, the message digest for OI (OIMD), and the customer's public key, then the bank can compute

$$\mathbf{H(H[OI]||OIMD);\ D(\mathit{PUc}, DS)}$$



Construction of Dual Signature

Again, if these two quantities are equal, then the bank has verified the signature. In summary,
1. The merchant has received OI and verified the signature.
2. The bank has received PI and verified the signature.
3**.** The customer has linked the OI and PI and can prove the linkage.

For example, suppose the merchant wishes to substitute another OI in this transaction, to its advantage. It would then have to find another OI whose hash matches the existing OIMD. With SHA-1, this is deemed not to be feasible. Thus, the merchant cannot link another OI with this PI.

UNIT-V
Assignment-Cum-Tutorial Questions
SECTION-A

Objective Questions

1. Modification of user data comes under_________ threat [ ]

a) Integrity b) confidentiality c)Denial of service d)authentication

2. Cryptography checksum is the counter measure of _____threat. [ ]

a) confidentiality b) denial of service c) integrity d) authentication

3. SSL is the security provided at_________level. [ ]

a) network layer b) transport layer

c) application layer d) presentation layer

4. TLS is the security provided at_________level. [ ]

a) network layer b) transport layer

c) application layer d) presentation layer

5. The Major version of SSL is ____________bytes [ ]

a) 210 b)212 c) 214 d) 216

6. The max allowable MAC length in SSL record protocol is ______ [ ]

a) 28 bytes b) 210 bytes c) 29 bytes d) 211 bytes

7. Change cipher spec protocol uses _______bytes [ ]

a) 2 b)3 c) 4 d) 1

8. SSL was originated by________ [ ]

a) internet explorer b) netscape

c) real player d) any browser

9. SSL was designed to provide a reliable ________secure service. [ ]

a) one-to-one b) one-to-many

c) many-to-one d) many-to-many

10. A connection of transport layer provides [ ]
a) A suitable type of service between client and server
b) An association between client and server
c) Both (a) and(b)
d) Neither (a) nor (b)

11. A session is a transport provides [ ]
a) A suitable type of service between client and server
b) An association between client and server
c) Both (a) and(b)
d) Neither (a) nor (b)

12. Dual signature is used to link___________ [ ]
a) Two messages intended for two same recipients
b) Two messages intended for two different recipients
c) one message intended for two same recipients
d) one message intended for two different recipients

13. Payment processing includes___________ [ ]
1. Purchase request 2.Payment authorization 3.Payment capture
a) Both 1 and 2
b) Both 2 and 3
c) Both 1 and
d) All of the above

14. In the construction of dual signature, it uses____ hash function [ ]
a) SHA b) SHA-1 c) RSA d) IDEA

15. SET is an open encryption ,designed to protect ____on internet [ ]
a) Debit card transaction c) credit card transaction
b) Account transfers d) Deposits

16. This is not a key feature of SET [ ]
a) confidentiality b) non-repudiation
c) integrity d) authentication

17. This is not a higher protocol defined as a part of SSL [ ]
a) Handshake protocol c) Change cipher spec protocol
b) Alert protocol d) FTP

18. Cipher spec uses________ [ ]
a) Data Encryption algorithm
b) Hash algorithm used for MAC calculation
c) Cryptographic attributes such as hash size

d) All of the above

19. Book1 of SET application gives________ [ ]

a) Business description c) Programmer's guide

b) Formal protocol Definition d) All of the above

SECTION-B

SUBJECTIVE QUESTIONS

1. List and briefly define the business requirements for secure payment processing with credit cards over the internet?
2. List and briefly define the principle categories of SET participants.
3. What are various web threats?
4. What is SSL? Explain the features of SSL.
5. Explain key features of SET.
6. What is the use of SSL protocol? Explain SSL record protocol operation with SSL record format.
7. Write and explain TLS functions and alert codes of Transport Layer Security
8. Describe about SSL secure communication and SSL authentication.
9. Describe in general how online payment processing is done.
10. What does SSL handshake establish? How is it performed?
11. Differentiate between SSLV3 and TLS.
12. Write a short note on dual signature.

INFORMATION SECURITY

UNIT VI

Learning Material

Unit – VI

Objectives:

Syllabus: System security

Intruders, Viruses and related threats, Firewall Design principles, Trusted Systems, Intrusion Detection Systems.

Outcomes:

Students will be able to

1. be able to identify some of the factors driving the need firewalls and trusted systems.
2. be able to identify intruders, viruses and related threats.
3. to identify intrusion detection systems and their signatures.

INTRUDER

One of the two most publicized threats to security is the intruder (the other is viruses), often referred to as a hacker or cracker.Three classes of intruders:

• **Masquerader:** An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account

• **Misfeasor:** A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges

• **Clandestine user:** An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit
collection

The masquerader is likely to be an outsider; the misfeasor generally is an insider;
and the clandestine user can be either an outsider or an insider.

Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system.

The following examples of intrusion:

• Performing a remote root compromise of an e-mail server

• Defacing a Web server

• Guessing and cracking passwords

• Copying a database containing credit card numbers

• Viewing sensitive data, including payroll records and medical information, without authorization

• Running a packet sniffer on a workstation to capture usernames and passwords

• Using a permission error on an anonymous FTP server to distribute pirated software and music files

• Dialing into an unsecured modem and gaining internal network access

• Posing as an executive, calling the help desk, resetting the executive's e-mail password, and learning the new password

• Using an unattended, logged-in workstation without permission

Intrusion Techniques

The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. Most initial attacks use system or software vulnerabilities that allow a user to execute code that opens a back door into the system.

Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it and learn passwords. The password file can be protected in one oftwo ways:

• **One-way function**: The system stores only the value of a function based on the

user's password. When the user presents a password, the system transforms that password and compares it with the stored value. In practice, the system usually performs a one-way transformation (not reversible) in which the password is used to generate a key for the one-way function and in which a fixed-length output is produced.

• **Access control:** Access to the password file is limited to one or a very few accounts.

If one or both of these countermeasures are in place, some effort is needed for a potential intruder to learn passwords. On the basis of a survey of the literature and interviews with a number of password crackers, [ALVA90] reports the following techniques for learning passwords:

1. Try default passwords used with standard accounts that are shipped with the
system. Many administrators do not bother to change these defaults.
2. Exhaustively try all short passwords (those of one to three characters).
3. Try words in the system's online dictionary or a list of likely passwords. Examples
of the latter are readily available on hacker bulletin boards.
4. Collect information about users, such as their full names, the names of their spouse and children, pictures in their office, and books in their office that are related to hobbies.
5. Try users' phone numbers, Social Security numbers, and room numbers.
6. Try all legitimate license plate numbers for this state.
7. Use a Trojan horse (described in Chapter 10) to bypass restrictions on access.
8. Tap the line between a remote user and the host system.

Password Protection

The front line of defense against intruders is the password system. Virtually all multiuser systems require that a user provide not only a name or identifier (ID) but also a password. The password serves to authenticate the ID of the individual logging on to the system. In turn, the ID provides security in the following ways:

• The ID determines whether the user is authorized to gain access to a system. In some systems, only those who already have an ID filed on the system are allowed to gain access.
• The ID determines the privileges accorded to the user. A few users may have supervisory or "superuser" status that enables them to read files and perform functions that are especially protected by the operating system. Some systems

have guest or anonymous accounts, and users of these accounts have more limited privileges than others.

• The ID is used in what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

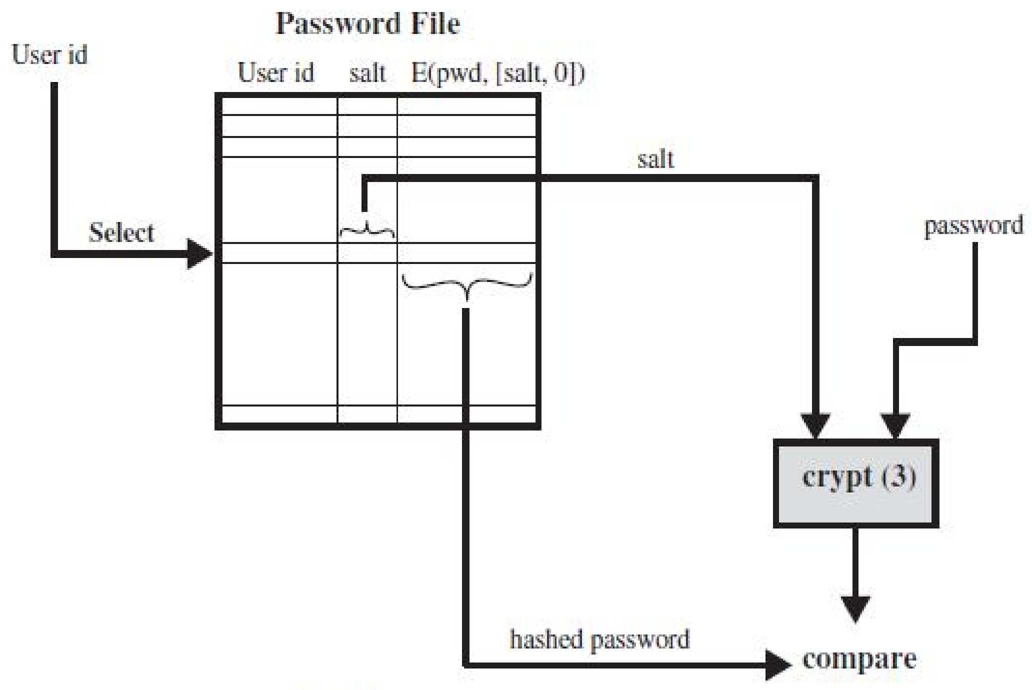
THE VULNERABILITY OF PASSWORDS To understand the nature of the threat to password-based systems, let us consider a scheme that is widely used on UNIX, in which passwords are never stored in the clear. Each user selects a password of up to eight printable characters in length. This is converted into a 56-bit value (using 7-bit ASCII) that serves as the key input to an encryption routine. The encryption routine, known as crypt(3), is based on DES. The DES algorithm is modified using a 12-bit “salt” value. Typically, this value is related to the time at which the password is assigned to the user. The modified DES algorithm is exercised with a data input consisting of a 64-bit block of zeros. The output of the algorithm then serves as input for a second encryption. This process is repeated for a total of 25 encryptions. The resulting 64-bit
output is then translated into an 11-character sequence. The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID. This method has been shown to be secure against a variety of cryptanalytic attacks.

The salt serves three purposes:

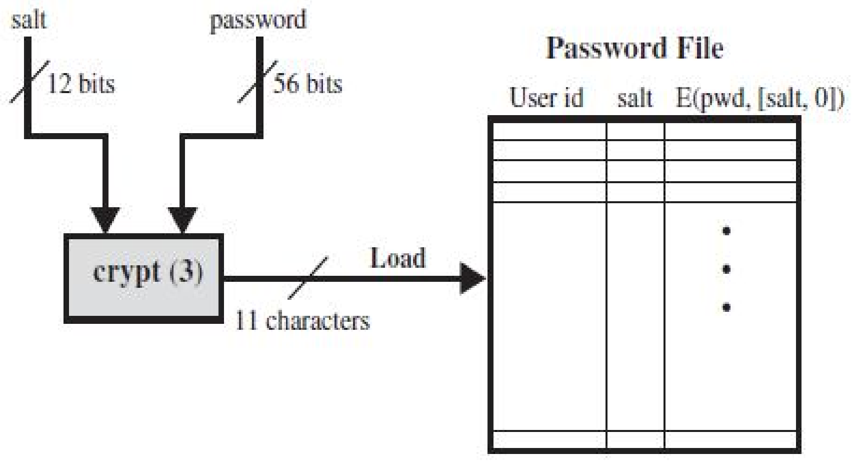
• It prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned at different times. Hence, the “extended” passwords of the two users will differ.

• It effectively increases the length of the password without requiring the user to remember two additional characters. Hence, the number of possible passwords is increased by a factor of 4096, increasing the difficulty of guessing a password.

• It prevents the use of a hardware implementation of DES, which would ease the difficulty of a brute-force guessing attack.



(b) Verifying a password



(a) Loading a new password

UNIX Password Scheme

Password Selection Strategies

Many users choose a password that is too short or too easy to guess. At the other extreme, if users are assigned passwords consisting of eight randomly selected printable characters, password cracking is effectively impossible. But it would be almost as impossible for most users to remember their passwords. Fortunately, even if we limit the password universe to strings of characters that are reasonably memorable, the size of the universe is still too large to permit practical cracking. Our goal, then, is to eliminate guessable passwords while allowing the user to select a password that is memorable. Four basic techniques are in use:

• User education

• Computer-generated passwords

• Reactive password checking

• Proactive password checking

Users can be told the importance of using hard-to-guess passwords and can be provided with guidelines for selecting strong passwords. This **user education** strategy is unlikely to succeed at most installations, particularly where there is a large user population or a lot of turnover. Many users will simply ignore the guidelines. Others may not be good judges of what is a strong password. For example, many users (mistakenly) believe that reversing a word or capitalizing the last letter makes a password unguessable.

Computer-generated passwords also have problems. If the passwords are quite random in nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down. In general, computer-generated password schemes have a history of poor acceptance by users. FIPS PUB 181 defines one of the best-designed automated password generators. The standard includes not only a description of the approach but also a complete listing of the C source code of the algorithm. The algorithm generates words by forming pronounceable syllables and concatenating them to form a word. A random number generator produces a random stream of characters used to construct the syllables and words.

A **reactive password checking** strategy is one in which the system periodically runs its own password cracker to find guessable passwords. The system cancels any passwords that are guessed and notifies the user. This tactic has a number of drawbacks. First, it is resource intensive if the job is done right. Because a determined opponent who is able to steal a password file can devote full CPU time to the task for hours or even days, an effective reactive password checker is at a distinct disadvantage. Furthermore, any existing passwords remain vulnerable until the reactive password checker finds them.

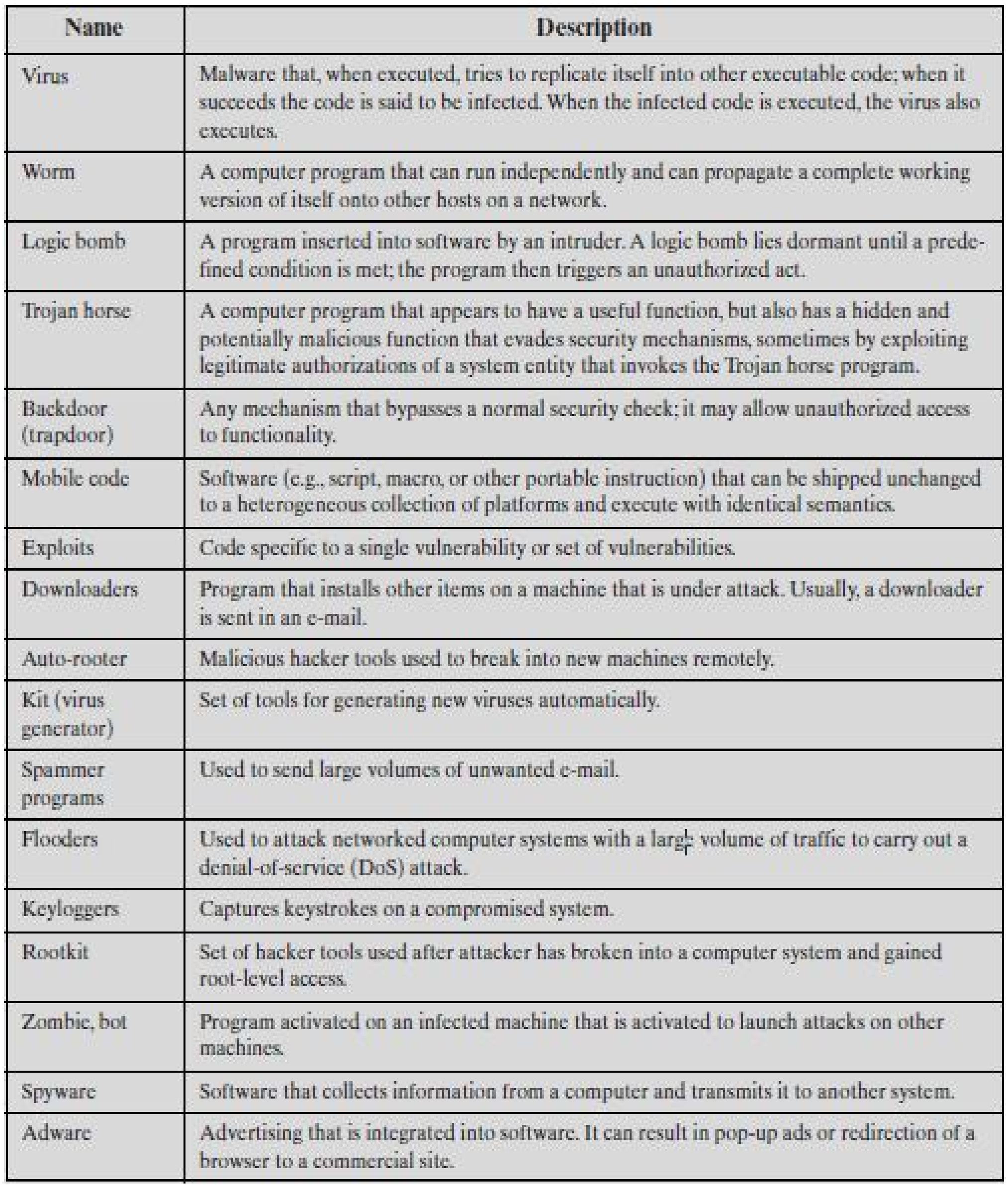
The most promising approach to improved password security is a **proactive password checker**. In this scheme, a user is allowed to select his or her own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Such checkers are based on the philosophy that, with sufficient guidance from the system, users can select memorable passwords from a fairly large password space that are not likely to be guessed in a dictionary attack.

VIRUSES AND RELATED THREATS

- Malicious software is software that is intentionally included or inserted in a system for a harmful purpose.
- A virus is a piece of software that can “infect” other programs by modifying
  them; the modification includes a copy of the virus program, which can then go on to infect other programs.
- A worm is a program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm

may be activated to replicate and propagate again. In addition to propagation,the worm usually performs some unwanted function.

- A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from using that service.
- A distributed denial of service attack is launched from multiple coordinated sources.



Name	Description
Virus	Malware that, when executed, tries to replicate itself into other executable code; when it succeeds the code is said to be infected. When the infected code is executed, the virus also executes.
Worm	A computer program that can run independently and can propagate a complete working version of itself onto other hosts on a network.
Logic bomb	A program inserted into software by an intruder. A logic bomb lies dormant until a predefined condition is met; the program then triggers an unauthorized act.
Trojan horse	A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes the Trojan horse program.
Backdoor (trapdoor)	Any mechanism that bypasses a normal security check; it may allow unauthorized access to functionality.
Mobile code	Software (e.g., script, macro, or other portable instruction) that can be shipped unchanged to a heterogeneous collection of platforms and execute with identical semantics.
Exploits	Code specific to a single vulnerability or set of vulnerabilities.
Downloaders	Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious hacker tools used to break into new machines remotely.
Kit (virus generator)	Set of tools for generating new viruses automatically.
Spammer programs	Used to send large volumes of unwanted e-mail.
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial-of-service (DoS) attack.
Keyloggers	Captures keystrokes on a compromised system.
Rootkit	Set of hacker tools used after attacker has broken into a computer system and gained root-level access.
Zombie, bot	Program activated on an infected machine that is activated to launch attacks on other machines.
Spyware	Software that collects information from a computer and transmits it to another system.
Adware	Advertising that is integrated into software. It can result in pop-up ads or redirection of a browser to a commercial site.

Terminology of Malicious Programs

Types of Viruses

- **Parasitic Virus** - attaches itself to executable files as part of their code. Runs whenever the host program runs.
- **Memory-resident Virus** - Lodges in main memory as part of the residual operating system.
- **Boot Sector Virus** - infects the boot sector of a disk, and spreads when the operating system boots up (original DOS viruses).
- **Stealth Virus** - explicitly designed to hide from Virus Scanning programs.
- **Polymorphic Virus** - mutates with every new host to prevent signature detection.

FIREWALL DESING PRINCIPLES

A firewall forms a barrier through which the traffic going in each direction must pass. A firewall security policy dictates which traffic is authorized to pass in each direction. A firewall may be designed to operate as a filter at the level of IP packets, or may operate at a higher protocol layer

Firewall Characteristics

- All traffic from inside to outside must pass through the firewall (physically blocking all access to the local network except via the firewall)
- Only authorized traffic (defined by the local security police) will be allowed to pass
- The firewall itself is immune to penetration (use of trusted system with a secure operating system)

Four general techniques:

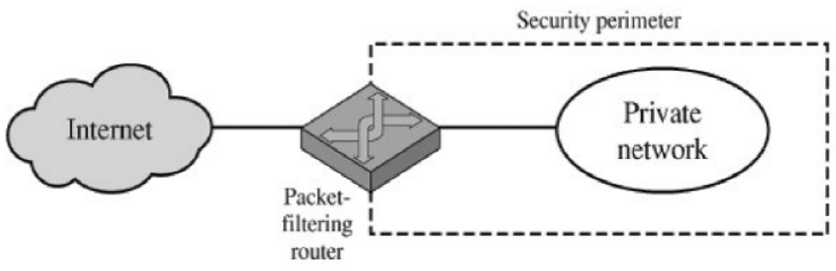
- Service control :Determines the types of Internet services that can be accessed, inbound or outbound

- Direction control: Determines the direction in which particular service requests are allowed to flow
- User control: Controls access to a service according to which user is attempting to access it
- Behavior control: Controls how particular services are used (e.g. filter e-mail)

Types of Firewalls: Three common types of Firewalls

- Packet-filtering routers
- Application-level gateways
- Circuit-level gateways

Packet- filtering routers



(a) Packet-filtering router

- Applies a set of rules to each incoming IP packet and then forwards or discards the packet
- Filter packets going in both directions
- The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header
- Two default policies (discard orforward)

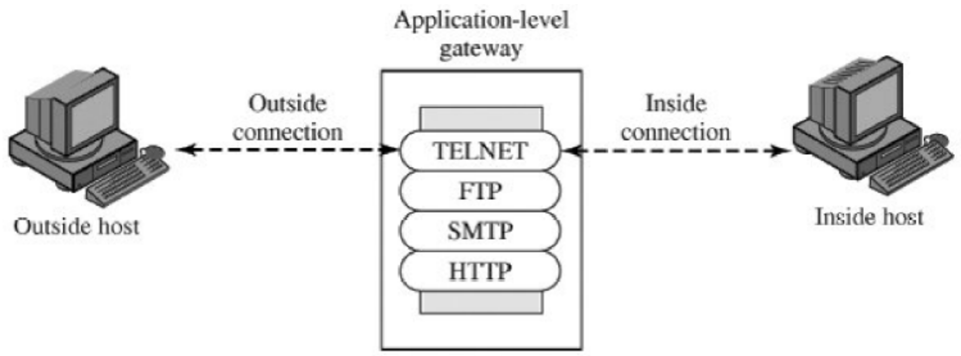
Advantages:

- Simplicity
- Transparency to users
- High speed

Disadvantages:

- Difficulty of setting up packet filter rules
- Lack of Authentication

Application-Level Gateways



(b) Application-level gateway

Application-level Gateway

- Also called proxy server
- Acts as a relay of application-level traffic

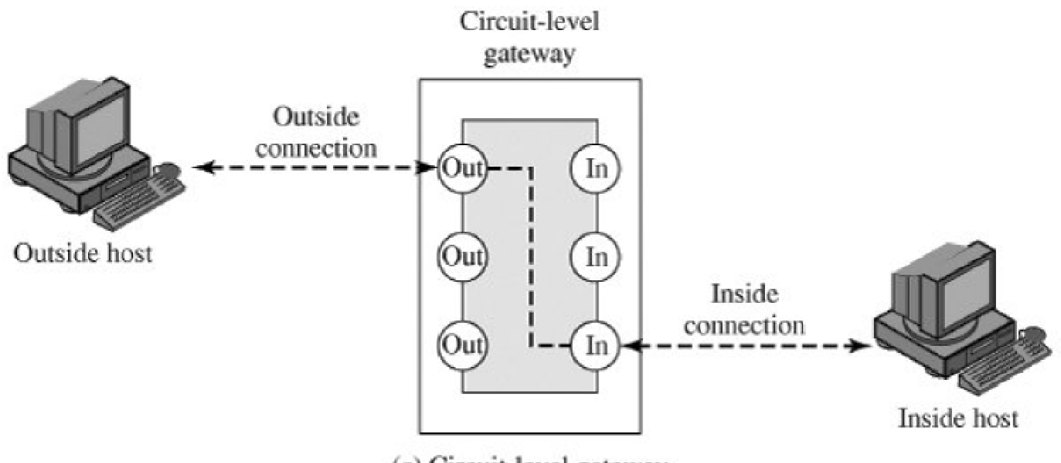
Advantages:

- Higher security than packet filters
- Only need to scrutinize a few allowable applications
- Easy to log and audit all incoming traffic

Disadvantages:

- Additional processing overhead on each connection (gateway as splice point)
- What about not supported protocols?

Circuit-Level Gateways



(c) Circuit-level gateway

Circuit-level Gateway

- Stand-alone system or
- Specialized function performed by an Application-level Gateway
- Sets up two TCP connections
- 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
- Typically use is a situation in which the system administrator trusts the internal users
- An example is the SOCKS package

Bastion Host

- A system identified by the firewall administrator as a critical strong point in the network's security

- The bastion host serves as a platform for an application-level or circuit-level gateway

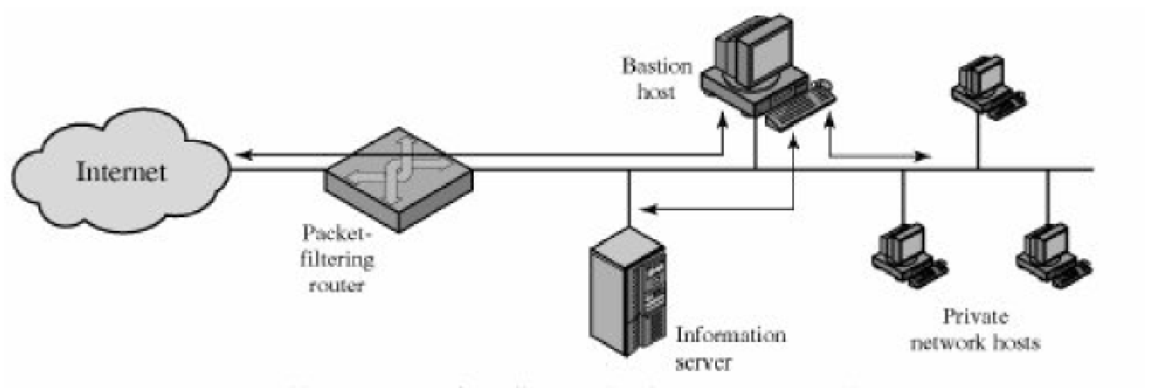
Firewall Configurations

In addition to the use of simple configuration of a single system (single packet filtering router or single gateway), more complex configurations are possible.

Three common configurations

- Screened host firewall system (singlehomed bastion host)
- Screened host firewall system (dualhomed bastion host)
- Screened-subnet firewall system

Screened host firewall system (singlehomed bastion host)



(a) Screened host firewall system (single-homed bastion host)

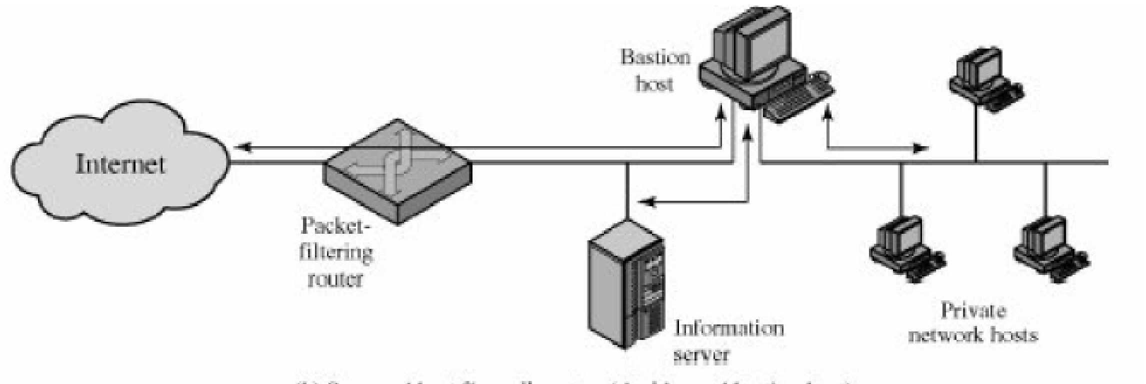
Firewall consists of two systems

- A packet-filtering router
- A bastion host

- Configuration for the packet-filtering router:
  Only packets from and to the bastion host are allowed to pass through the router
- The bastion host performs authentication and proxy functions

- This configuration also affords flexibility in providing direct Internet access (public information server, e.g. Web server)

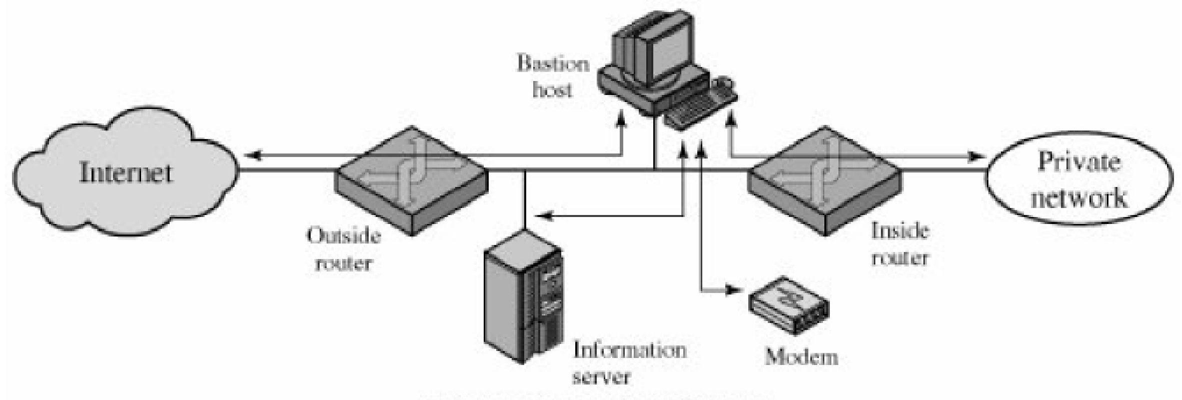
Screened host firewall system (dualhomed bastion host)



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

- The packet-filtering router is not completely compromised
- Traffic between the Internet and other hosts on the private network has to flow through the bastion host

Screened-subnet firewall system



(c) Screened-subnet firewall system

- Most secure configuration of the three
- Two packet-filtering routers are used

- Creation of an isolated sub-network

  Advantages:
- Three levels of defense to thwart intruders
- The outside router advertises only the existence of the screened subnet to the Internet (internal network is invisible to the Internet)
- The inside router advertises only the existence of the screened subnet to the internal network (the systems on the inside network cannot construct direct routes to the Internet)

TRUSTED SYSTEMS

One way to enhance the ability of a system to defend against intruders and malicious programs is to implement trusted system technology

Data Access Control

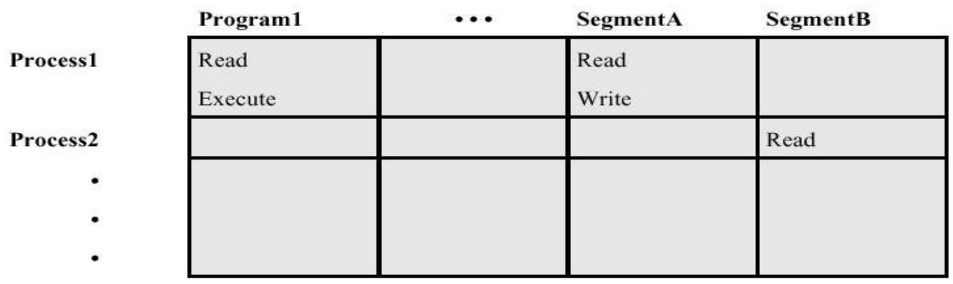
- Through the user access control procedure (log on), a user can be identified to the system
- Associated with each user, there can be a profile that specifies permissible operations and file accesses
- The operation system can enforce rules based on the user profile
- General models of access control:

  Access matrix

  Access control list

  Capability list

  Access matrix



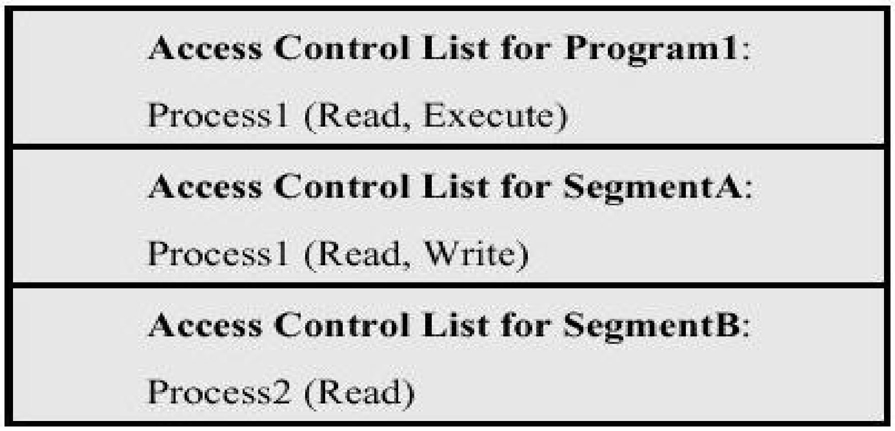
	Program1	• • •	SegmentA	SegmentB
Process1	Read Execute		Read Write	
Process2				Read
• • •				

Access Matrix: Basic elements of the model

- Subject: An entity capable of accessing objects, the concept of subject equates with that of process
- Object: Anything to which access is controlled (e.g. files, programs)
- Access right: The way in which an object is accessed by a subject (e.g. read, write, execute)

Access control list

- Access Control List: Decomposition of the matrix by columns
- An access control list, lists users and their permitted access right
- The list may contain a default or public entry



Access Control List for Program1:
Process1 (Read, Execute)

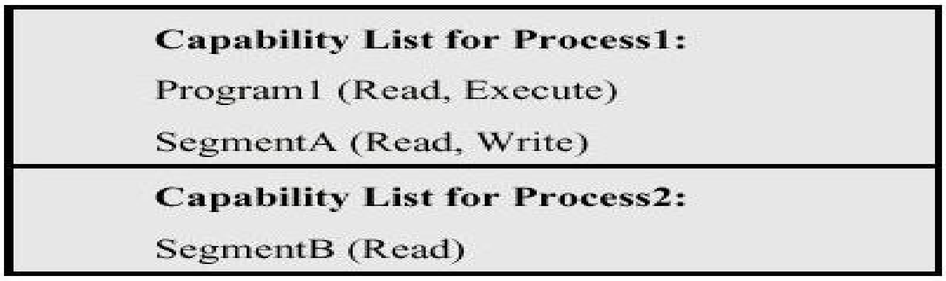
Access Control List for SegmentA:
Process1 (Read, Write)

Access Control List for SegmentB:
Process2 (Read)

Capability list

- Capability list: Decomposition of the matrix by rows
- A capability ticket specifies authorized objects and operations for a user

- Each user have a number of tickets



Capability List for Process1:
Program1 (Read, Execute)
SegmentA (Read, Write)

Capability List for Process2:
SegmentB (Read)

The Concept of Trusted System

- Protection of data and resources on thebasis of levels of security (e.g. military)
- Users can be granted clearances to access certain categories of data

Multilevel security

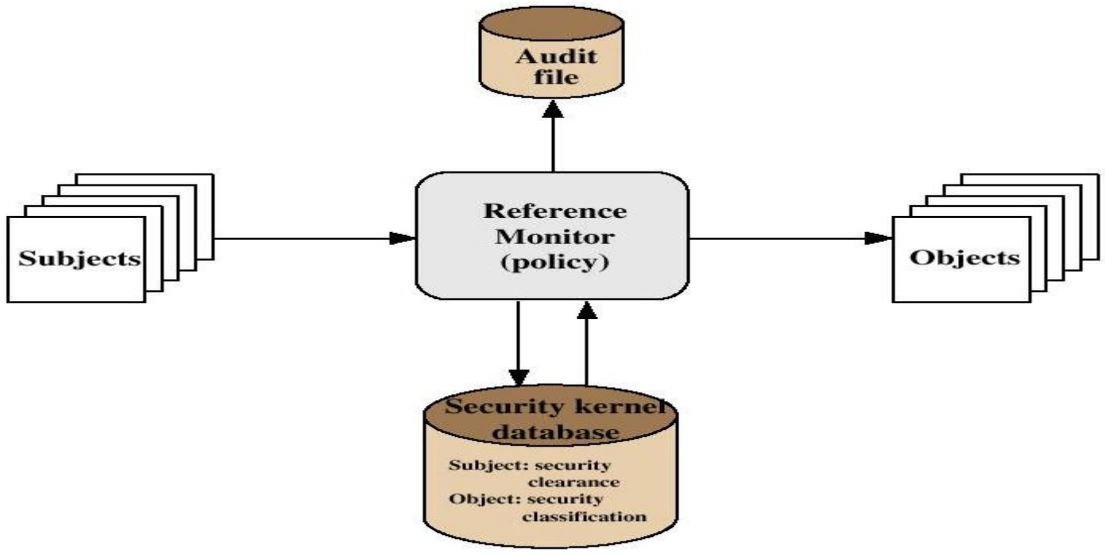
- Definition of multiple categories or levels of data

A multilevel secure system must enforce:

- No read up: A subject can only read an object of less or equal security level (Simple Security Property)
- No write down: A subject can only write into an object of greater or equal security level (*-Property)

Reference Monitor Concept: Multilevel

- security for a data processing system
- Controlling element in the hardware and operating system of a computer that regulates the access of subjects to objects on basis of security parameters
- The monitor has access to a file (security kernel database)
- The monitor enforces the security rules (no read up, no write down)


Audit file
Subjects
Reference Monitor (policy)
Objects
Security kernel database
Subject: security clearance
Object: security classification

Properties of the Reference Monitor

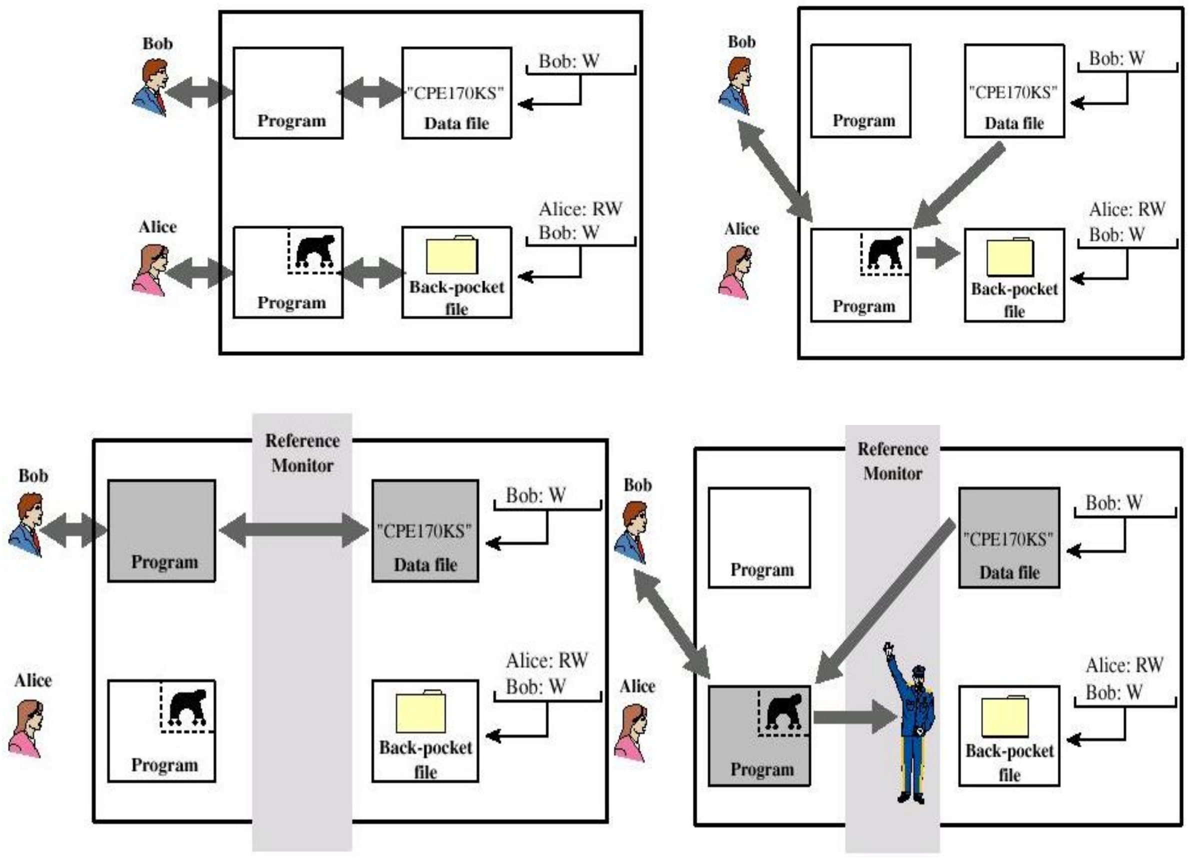
- Complete mediation: Security rules areenforced on every access
- Isolation: The reference monitor and database are protected from unauthorized modification
- Verifiability: The reference monitor's correctness must be provable (mathematically)

  A system that can provide such verifications (properties) is referred to as a trusted system

Trojan Horse Defense

- Secure, trusted operating systems are one way to increase the overall security in a system and can protect from many different attacks.

- As an example we will study how it can protect against a Trojan Horse attack.

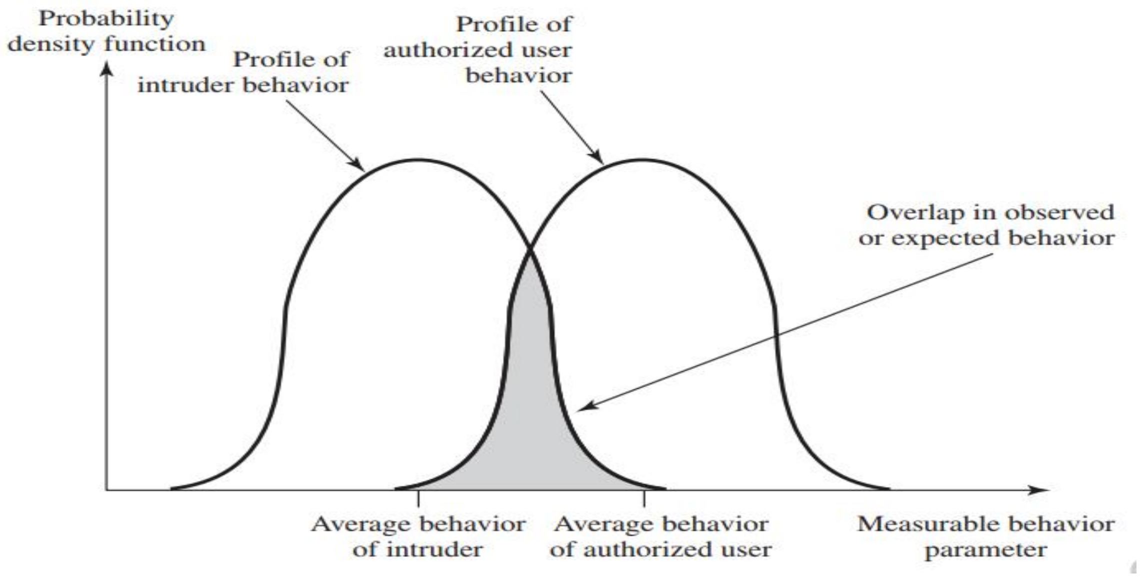


Trojan Horse and Secure Operating System

INTRUSION DETECTION SYSTEM

- If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to preempt the intruder, the sooner that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved.
- An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.

- Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.


Probability density function
Profile of intruder behavior
Profile of authorized user behavior
Overlap in observed or expected behavior
Average behavior of intruder
Average behavior of authorized user
Measurable behavior parameter

Profiles of Behavior of Intruders and Authorized Users

The following approaches to intrusion detection:

- **Statistical anomaly detection**: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior.

  a. Threshold detection: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.

  b. Profile based: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.

- **Rule-based detection**: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.

    a. Anomaly detection: Rules are developed to detect deviation from previous usage patterns.

    b. Penetration identification: An expert system approach that searches for suspicious behavior.

UNIT-VI

Assignment-Cum-Tutorial Questions

SECTION-A

Objective Questions

1. A person who is not authorized to use the system but penetrates it ______ [ ]

a) masquerade b) misfeasor c) clandestine user d) virus throatier

2. A person who seizes supervisory control of the system and uses this for some unauthorized purpose [ ]

a) masquerade b) misfeasor c) clandestine user d) virus throatier

3. The masquerade in general _____ [ ]

a) insider b) outsider c) a or b d) none of the above

4. Which of the following is not need host program [ ]

a) trapdoor b) worm c) logic bomb d) Trojan horse

5. Which of the following is not a phase of virus [ ]

a) propagation phase b) execution phase c) dedicated phase d) triggering phase

6. First generation antivirus approach uses__________ [ ]

a) heuristic scanners b) simple scanners c) activity traps d) full featured

7. Profile based detection comes under _______ [ ]

a) statistical anamoly detection system

b) rule-based detection system

c) timestamp method

d) detection-specific password scheme

8. Like biological virus, computer virus can__________ [ ]

a) detect human beings b) makes perfect copies of itself

c) kills human beings d) none of the above

9. ________ programs use network connection to spread from system to system [ ]

a) Trapdoor b) worm c) logic bomb d) Trojan horse

10. Arrange the phases in order: [ ]

I. propagation phase

II. execution phase

III. dedicated phase

a) I,II,III b) II,III,I c) I,III,II d) III,II,I

11. Virus places identical copy of itself in other programs in [ ]

a) Propagation phase b) execution phase c) dormant phase d) triggering phase

12. Packet filtering router applies rule on incoming ______ [ ]

a) IP packets b) IP Header c) IP address d) IP protocol field

13. The default actions taken by the packet filtering router are _ [ ]

a) Discard and release b) discard and forward c) forward and release

14. Which of the following is / are the types of firewall? [ ]

a) Packet Filtering Firewall

b) Dual Homed Gateway Firewall

c) Screen Host Firewall

d) All of the mentioned

15. What tells a firewall how to reassemble a data stream that has been divided into packets? [ ]
a) The source routing future
b) The number in the header's identification field
c) The destination IP address
d) The header checksum field in the packet header

SECTION-B

SUBJECTIVE QUESTIONS

1. Explain the different types of firewall and its configurations in detail.

2. What is intruder? Explain the types of Intrusion detection System in detail.

3. What is IDS? Explain the profile based IDS?

4. Briefly explain the following:

i) Trapdoors ii) Bacteria iii) Logic Bomb iv) Trojan horse

5. Explain various intrusion detection techniques.

6. What are the effects of malicious software? Write any two.

7. With a neat diagram explain the working principle of packet-filtering router?

8. What is virus? Explain different antivirus approaches

9. What is the difference between statistical anomaly detection and the rule-based intrusion detection?

10. Write a short notes on the following

a) Trojan horse attack. b) characteristics of firewalls.