SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT - 1

SCS1316 - NETWORK SECURITY

UNIT-1 NETWORK SECURITY AND NUMBER THEORY BASICS

Syllabus:

Network security- Examples of security violations - Computer security concepts-confidentiality-Integrity-Availability-Accountability, Challenges of computer security-Hacking-Vulnerability-threats-attacks-passive attacks-types-Active attacks-types-Denial of service attacks-Model for network security. Modular arithmetic- Addition-Inverse divisibility- prime numbers-Euler's theorem-Fermat's theorem

1.Introduction:

With the introduction of the computer, the need for automated tools for protecting files and other information stored on the computer became evident. This is especially the case for a shared system, and the need is even more acute for systems that can be accessed over the Internet. The generic name for the collection of tools designed to protect data and to thwart hackers is **computer security**.

Network security: Introduction of distributed systems and the use of networks and communications facilities for carrying data between terminal user and computer and between computer and computer. Network security measures are needed to protect data during their transmission. The term network security in general refers to internet security.

1.1 EXAMPLES OF SECURITY VIOLATIONS

1. User A transmits a file to user B. The file contains sensitive information (e.g., payroll records) that is to be protected from disclosure. User C, who is not authorized to read the file, is able to monitor the transmission and capture a copy of the file during its transmission.

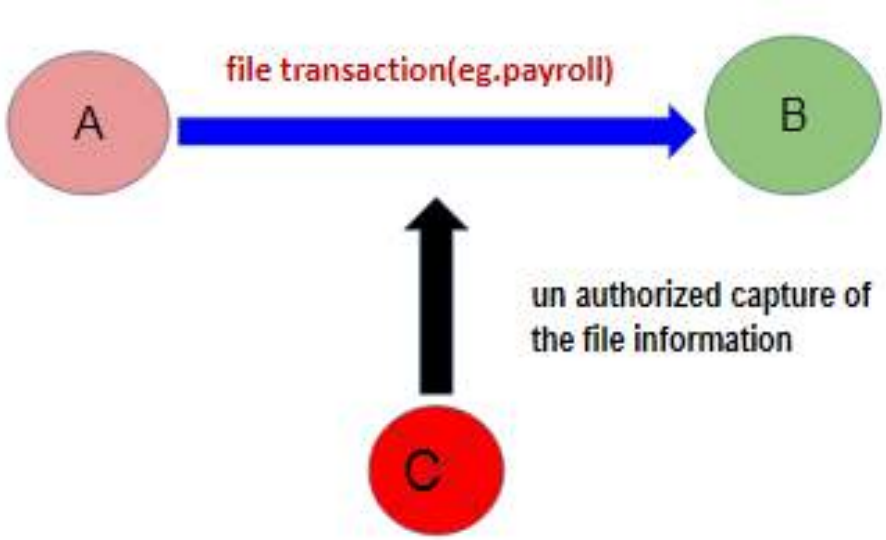


Fig 1.1 Example 1

2. A network manager, D, transmits a message to a computer, E, under its management. The message instructs computer E to update an authorization file to include the identities of a number of new users who are to be given access to that computer. User F intercepts the message, alters its contents to add or delete entries, and then forwards the message to E, which accepts the message as coming from manager D and updates its authorization file accordingly.

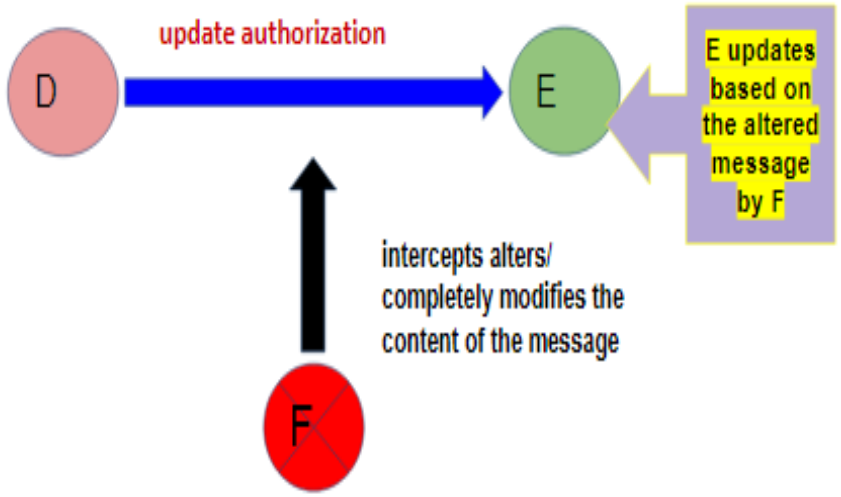


Fig 1.2 Example 2

3. Rather than intercept a message, user F constructs its own message with the desired entries and transmits that message to E as if it had come from manager D. Computer E accepts the message as coming from manager D and updates its authorization file accordingly.

4. A message is sent from a customer to a stockbroker with instructions for various transactions. Subsequently, the investments lose value and the customer denies sending the message.

Although this list by no means exhausts the possible types of security violations, it illustrates the range of concerns of network security.

1.2 COMPUTER SECURITY CONCEPTS

A Definition of Computer Security

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

1.2.1 COMPUTER SECURITY

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

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

1) **Confidentiality:** This term covers two related concepts:

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

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

2) **Integrity:** This term covers two related concepts:

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

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

3)Availability: Assures that systems work promptly and service is not denied to authorized users.

These three concepts form what is often referred to as the CIA triad (Figure 1.1). The three concepts embody the fundamental security objectives for both data and for information and computing services.

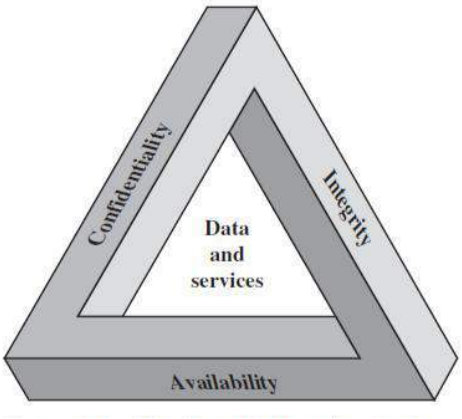


Fig 1.3 Security Requirements TRIAD

• **Confidentiality:** Preserving authorized restrictions on information access and disclosure, including means for protecting personal privacy and proprietary information. A loss of confidentiality is the unauthorized disclosure of information.
• **Integrity:** Guarding against improper information modification or destruction,including ensuring information nonrepudiation and authenticity. A loss of integrity is the unauthorized modification or destruction of information.

• **Availability:** Ensuring timely and reliable access to and use of information. A loss of availability is the disruption of access to or use of information or an information system.
Although the use of the CIA triad to define security objectives is well established, some in the security field feel that additional concepts are needed to present a complete picture. Two of the most commonly mentioned are,

• **Authenticity:** The property of being genuine and being able to be verified and trusted; confidence in the validity of a transmission, a message, or message originator. This means verifying that users are who they say they are and that each input arriving at the system came from a trusted source.

• **Accountability:** The security goal that generates the requirement for actions of an entity to be traced uniquely to that entity. This supports non-repudiation, deterrence, fault isolation, intrusion detection and prevention, and after-action recovery and legal action. Because truly secure systems are not yet an achievable goal, we must be able to trace a security breach to a responsible party. Systems must keep records of their activities to permit later forensic analysis

to trace security breaches or to aid in transaction disputes.

1.3 THE CHALLENGES OF COMPUTER SECURITY

Computer and network security is both fascinating and complex. Some of the reasons include:

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

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

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

4. Security mechanisms typically involve more than a particular algorithm or protocol. They also require that participants be in possession of some secret information (e.g.,

an encryption key), which raises questions about the creation, distribution, and protection of that secret information.

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

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

7. Security is still too often an afterthought to be incorporated into a system after the design is complete rather than being an integral part of the design process.

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

1.4 VULNERABILITY AND HACKING

Vulnerability

In computer security, a vulnerability is a weakness which can be exploited by a threat actor, such as an attacker, to cross privilege boundaries (i.e. perform unauthorized actions) within a computer system. To exploit vulnerability, an attacker must have at least one applicable tool or technique that can connect to a system weakness. In this frame, vulnerabilities are also known as the attack surface.

Hacking

Hacking is an attempt to exploit a computer system or a private network inside a computer. Simply put, it is the unauthorised access to or control over computer network security systems for some illicit purpose. One can easily assume them to be intelligent and highly skilled in computers.

1.5 SECURITY ATTACKS

Threat

A potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger that might exploit vulnerability.

Attack

An assault on system security that derives from an intelligent threat.That is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

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

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

Passive Attacks

Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions as shown in Fig 1.4. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are the release of message contents and traffic analysis.

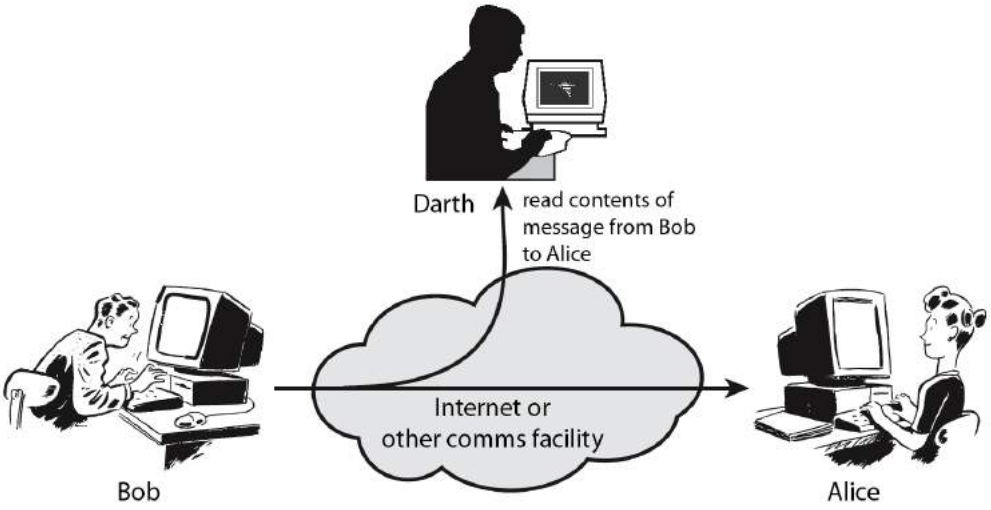


Fig 1.4 Passive Attacks

The release of message contents is easily understood (Figure 1.5). A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.

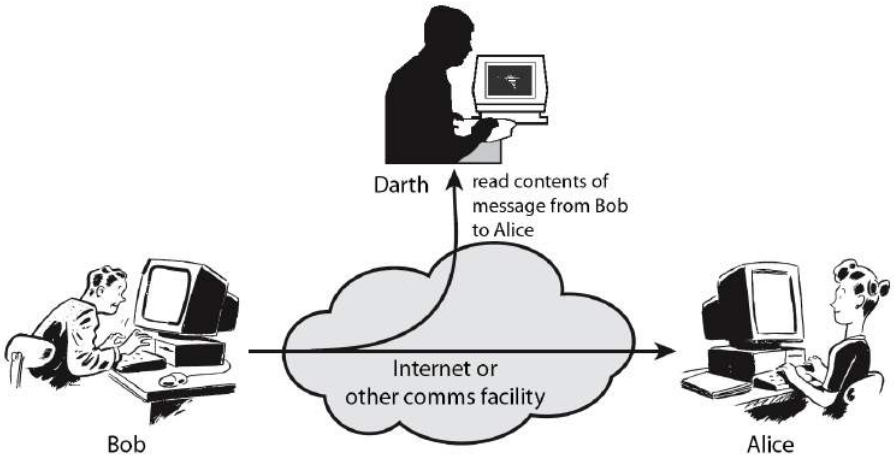


Fig 1.5 Release of Message

A second type of passive attack, traffic analysis, is subtler (Figure 1.6). Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption.

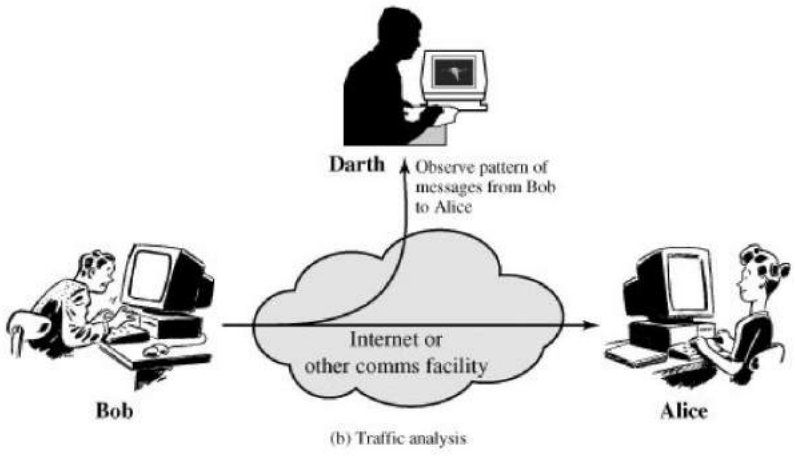


Fig 1.6 Traffic analysis

Passive attacks are very difficult to detect, because they do not involve any alteration of the data. Typically, the message traffic is sent and received in an apparently normal fashion, and neither the sender nor the 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. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

Active Attacks

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

A masquerade takes place when one entity pretends to be a different entity (Figure 1.7). A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.

Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect (Figure 1.8).

Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect (Figure 1.9). For example, a message meaning "Allow John Smith to read confidential file accounts" is modified to mean "Allow Fred Brown to read confidential file accounts."

The denial of service prevents or inhibits the normal use or management of communications facilities (Figure 1.10). This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is the disruption of an entire network—either by disabling the network or by overloading it with messages so as to degrade performance.

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

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

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

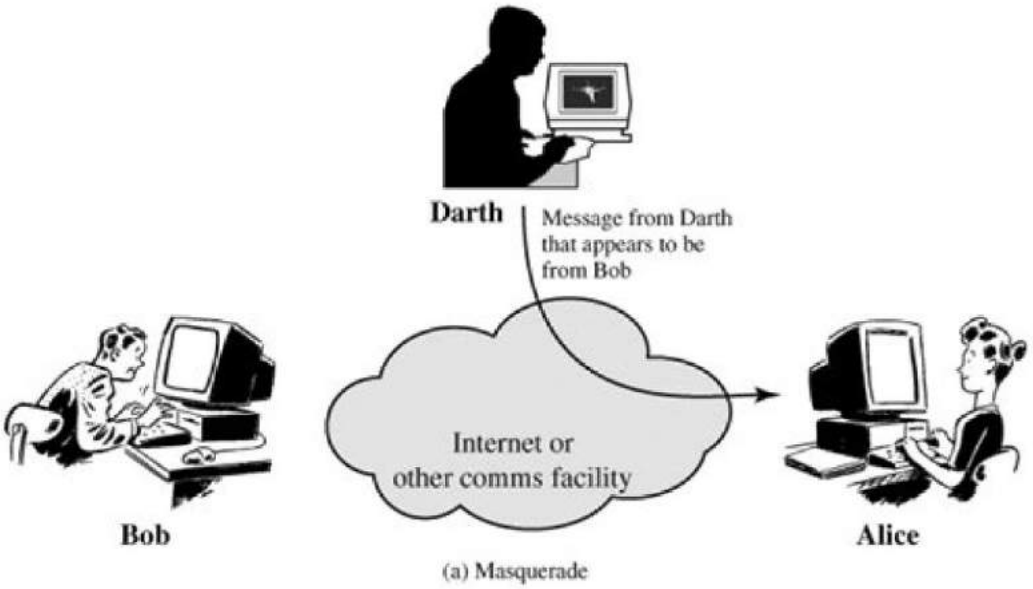


Fig 1.7 Masquerade

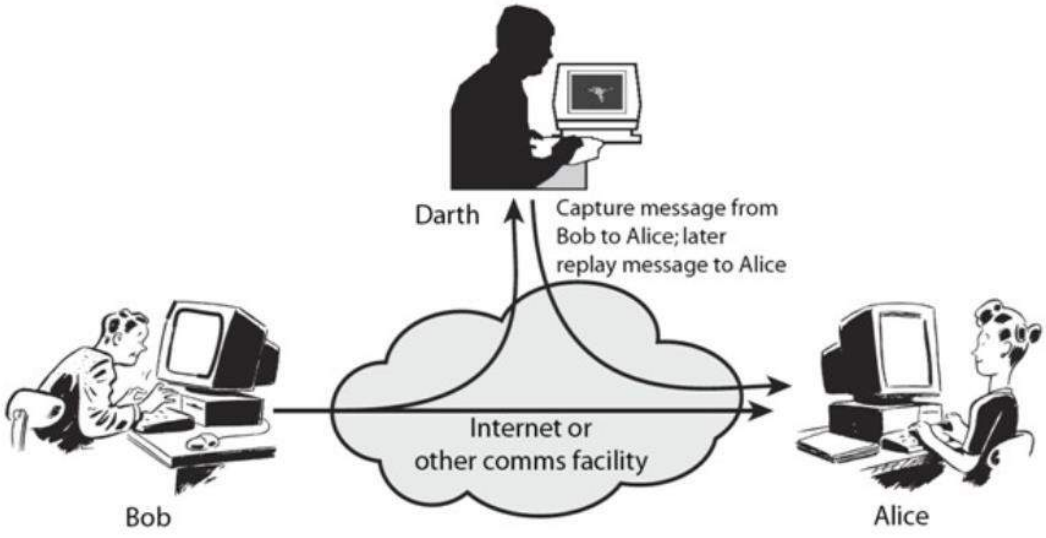


Fig 1.8 Replay

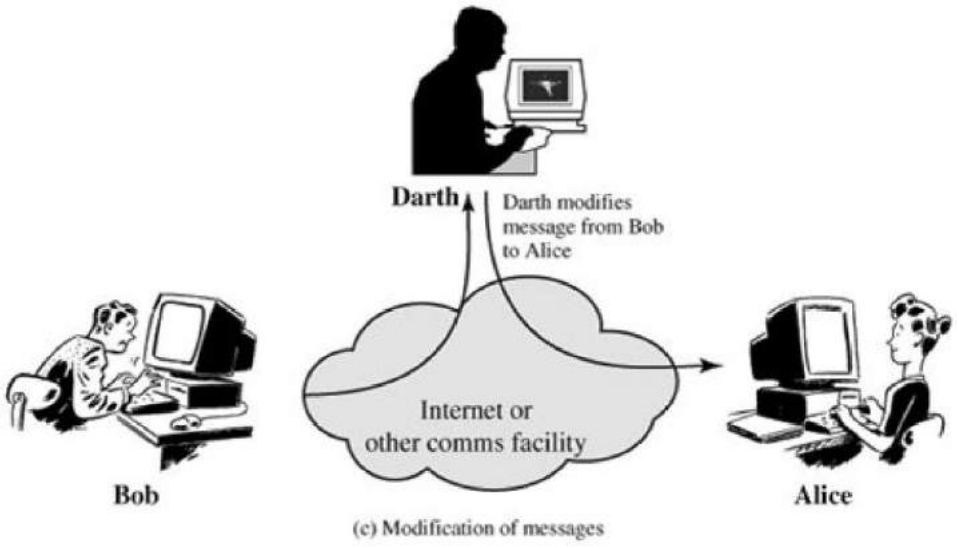


Fig 1.8 **Modification of messages**

Denial of Service Attacks

With a DoS attack, a hacker attempts to render a network or an Internet resource, such as a web server, worthless to users. A DoS attack typically achieves its goal by sending large amounts of repeated requests that paralyze the network or a server.

A common form of a DoS attack is a SYN flood, where the server is overwhelmed by embryonic connections. A hacker sends to a server countless Transmission Control Protocol (TCP) synchronization attempts known as SYN requests. The server answers each of those requests with a SYN ACK reply and allocates some of its computing resources to servicing this connection when it becomes a "full connection." Connections are said to be embryonic or half-opened until the originator completes the three-way handshake with an ACK for each request originated. A server that is inundated with half-opened connections soon runs out of resources to allocate to upcoming connection requests, thus the expression "denial of service attack."

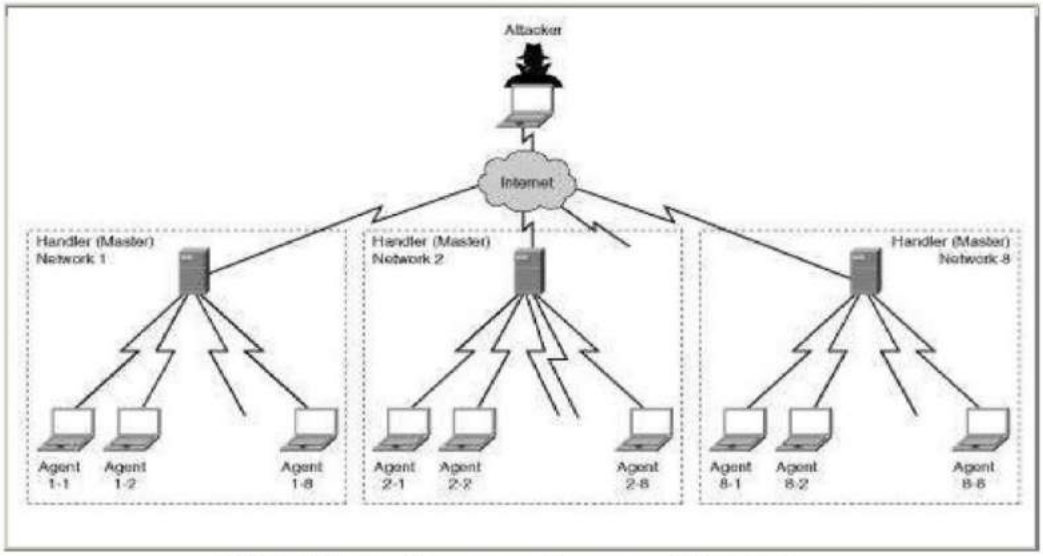
The following sidebars provide the anatomy of DoS attacks and distributed DoS (DDoS) attacks.

Anatomy of a Simple DoS Attack

A proverbial DoS attack called Land.c sends a TCP SYN request, giving the target host's address as both source and destination, and using the same port on the target host as both source and destination (for example, source address 10.0.0.1:139 to destination address 10.0.0.1:139).

Anatomy of a Complex Distributed DoS Attack

A common form of DoS attack is a DDoS attack. In the case of DDoS, an attacker finds hosts that he can compromise in different organizations and turns them into handlers by remotely installing DDoS handler software on them, as shown in fig



DDoSCreating an Army of Agents

Fig 1.9 DOS

Those handlers in turn scan their own corporate network, hunting for workstations to compromise and turn into DDoS agents. Those agents are also referred to as bots, thus the expression of botnets.

When his army of agents is strategically in place, the hacker launches the attack. He transmits his orders for the mission to the handlers and agents; these orders usually cause each of these hosts to send large quantities of packets to the same specific destination, at a precise time, thus overwhelming the victim and the path to it. It also creates significant congestion on corporate networks that are infected with handlers and agents when they all simultaneously launch their attack on the ultimate victim.

1.6 A MODEL FOR NETWORK SECURITY:

A Network Security Model exhibits how the security service has been designed over the network to prevent the opponent from causing a threat to the confidentiality or authenticity of the information that is being transmitted through the network.

For a message to be sent or receive there must be a sender and a receiver. Both the sender and receiver must also be mutually agreeing to the sharing of the message. Now, the transmission of a message from sender to receiver needs a medium i.e. Information channel which is an Internet service.

A logical route is defined through the network (Internet), from sender to the receiver and using the communication protocols both the sender and the receiver established communication.

Any security service would have the three components discussed below:

1. Transformation of the information which has to be sent to the receiver. So, that any opponent present at the information channel is unable to read the message. This indicates the encryption of the message.

It also includes the addition of code during the transformation of the information which will be used in verifying the identity of the authentic receiver.

2. Sharing of the secret information between sender and receiver of which the opponent must not any clue. Yes, we are talking of the encryption key which is used during the encryption of the message at the sender's end and also during the decryption of message at receiver's end.

3. There must be a trusted third party which should take the responsibility of distributing the secret information (key) to both the communicating parties and also prevent it from any opponent.

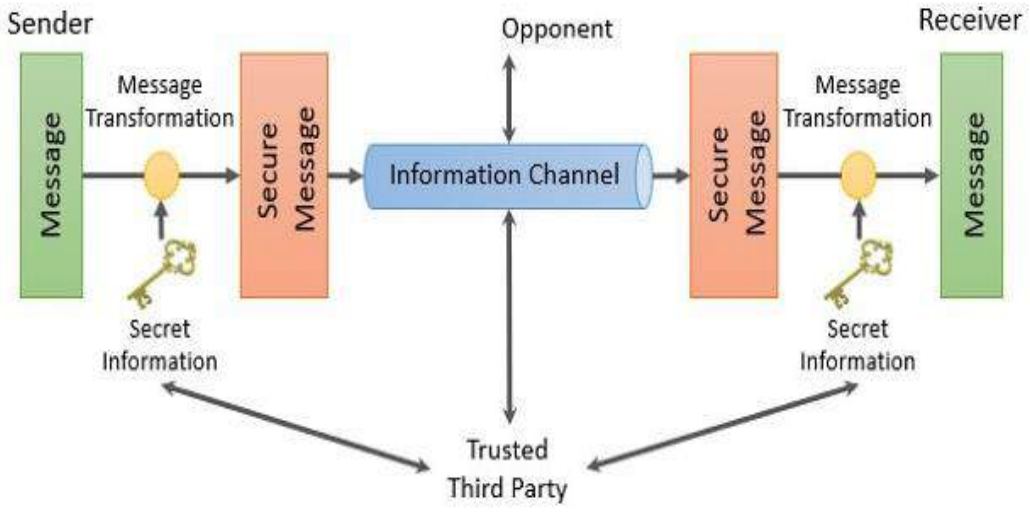


Fig 1.11 A Model for Network Security

The network security model presents the two communicating parties sender and receiver who mutually agrees to exchange the information. The sender has information to share with the receiver.

But sender cannot send the message on the information cannel in the readable form as it will have a threat of being attacked by the opponent. So, before sending the message through the information channel, it should be transformed into an unreadable format. Secret information is used while transforming the message which will also be required when the message will be retransformed at the recipient side. That's why a trusted third party is required which would take the responsibility of distributing this secret information to both the parties involved in communication.

So, considering this general model of network security, one must consider the following four tasks while designing the security model.

1. To transform a readable message at the sender side into an unreadable format, an appropriate algorithm should be designed such that it should be difficult for an opponent to crack that security algorithm.

2. Next, the network security model designer is concerned about the generation of the secret information which is known as a key.

This secret information is used in conjunction with the security algorithm in order to transform the message.

3. Now, the secret information is required at both the ends, sender's end and receiver's end. At sender's end, it is used to encrypt or transform the message into unreadable form and at the receiver's end, it is used to decrypt or retransform the message into readable form. So, there must be a trusted third party who will distribute the secret information to both sender and receiver. While designing the network security model designer must also concentrate on developing the methods to distribute the key to the sender and receiver. An appropriate methodology must be used to deliver the secret information to the communicating parties without the interference of the opponent.
It is also taken care that the communication protocols that are used by the communicating parties should be supporting the security algorithm and the secret key in order to achieve the security service.

1.7 NETWORK ACCESS SECURITY MODEL

Network access security model which is designed to secure the information system which can be accessed by the attacker through the network.
Attackers who attack your system that is accessible through the internet. These attackers fall into two categories:

1. Hacker: The one who is only interested in penetrating into your system. They do not cause any harm to your system they only get satisfied by getting access to your system.
2. Intruders: These attackers intend to do damage to your system or try to obtain the information from the system which can be used to attain financial gain.

The attacker can place a logical program on your system through the network which can affect the software on your system. This leads to two kinds of risks:
a. Information threat: This kind of threats modifies data on the user's behalf to which actually user should not access. Like enabling some crucial permission in the system.
b. Service threat: This kind of threat disables the user from accessing data on the system.

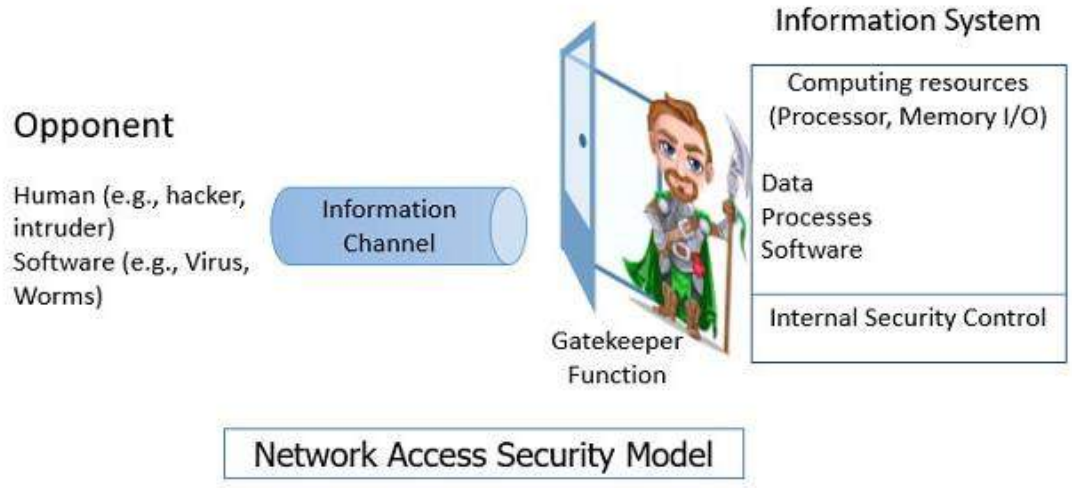


Fig 1.12 Network access security model

There are two ways to secure your system from attacker of which the first is to introduce the gatekeeper function. Introducing gatekeeper function means introducing login-id and passwords which would keep away the unwanted access.
In case the unwanted user gets access to the system the second way to secure your system is introducing internal control which would detect the unwanted user trying to access the system by analyzing system activities. This second method we call as antivirus which we install on our system to prevent the unwanted user from accessing your computer system through the internet.

1.8 MODULAR ARITHMETIC

Modulo, means remainder. Modulo arithmetic is the arithmetic of remainders.

If any integer a can be expressed as a = b+kn then in modulo arithmetic it can be stated as a mod n = b. F or example a=33 and n=5 then 33 mod 5= 3. (should be read as 3 mod 5)

This can be obtained by successive subtraction of n from a. In the above example the successive subtraction is as shown below.

1.8.1 The quotient remainder theorem

- To prove some properties about modular arithmetic we often make use of the quotient remainder theorem.
- It is a simple idea that comes directly from long division.

  The quotient remainder theorem says:

Given any integer A, and a positive integer B, there exist unique integers Q and R such that

$A = B * Q + R$ where $0 \leq R < B$

When we divide A by B in long division, Q is the quotient and R is the remainder.

i.e A -DIVIDEND

B-DIVISOR /MODULUS

Q-QUOTIENT

R-REMINDER/ RESIDUE

If we can write a number in this form then A mod B = R

Examples

A = 7, **B = 2**

7 = **2** * 3 + **1**
7 mod **2** = **1**

A = 8, **B = 4**

8 = **4** * 2 + **0**
8 mod **4** = **0**

A = 13, **B = 5**

13 = **5** * 2 + **3**
13 mod **5** = **3**

A = -16, **B = 26**

-16 = **26** * -1 + **10**
-16 mod **26** = **10**

1.8.2 Modular addition and subtraction

(A + B) mod C = (A mod C + B mod C) mod C

Example:

Let **A=14, B=17, C=5**

Let's verify: **(A + B) mod C = (A mod C + B mod C) mod C**
LHS = Left Hand Side of the Equation
RHS = Right Hand Side of the Equation

LHS = (A + B) mod C
LHS = (**14 + 17**) mod 5
LHS = **31** mod 5
LHS = 1

RHS = (A mod C + B mod C) mod C
RHS = (**14 mod 5 + 17 mod 5**) mod 5
RHS = (**4 + 2**) mod 5
RHS = 1

LHS = RHS = 1

1.8.3 Multiplication

(A * B) mod C = (A mod C * B mod C) mod C

Example for Multiplication:

Let **A=4, B=7, C=6**

Let's verify: **(A * B)** mod C = (**A mod C** * **B mod C**) mod C

LHS= Left Hand Side of the Equation

RHS= Right Hand Side of the Equation

LHS = **(A * B)** mod C

LHS = **(4 * 7)** mod 6

LHS = **28** mod 6

LHS = **4**

RHS = (**A mod C** * **B mod C**) mod C

RHS = (**4 mod 6** * **7 mod 6**) mod 6

RHS = (**4 * 1**) mod 6

RHS = **4 mod 6**

RHS = **4**

LHS = RHS = 4

1.8.4 Exponentiation

A^B mod C = ((A mod C)^B) mod C

Example

What is $3^{16} \pmod 4$?

We observe that

$$3^2 \equiv 9 \equiv 1 \pmod 4.$$

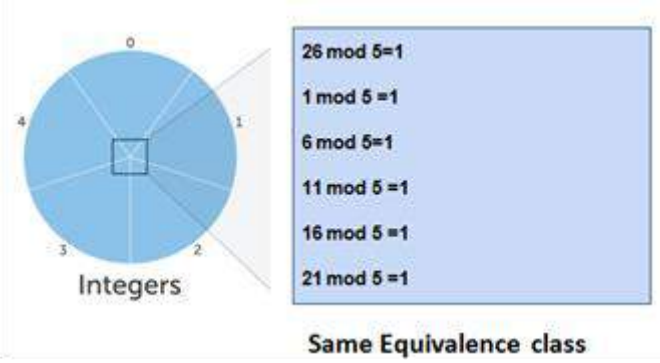
Then by the property of exponentiation, we have

$$\begin{aligned} 3^{16} \pmod 4 &\equiv \left(3^2\right)^8 \pmod 4 \\ &\equiv (1)^8 \pmod 4 \\ &\equiv 1 \pmod 4. \square \end{aligned}$$

1.9 CONGRUENCE MODULO

$$A \equiv B (\text{mod } C)$$

This says that A is congruent to B modulo C



1. $\equiv$ is the symbol for congruence, which means the values A and B are in the same **equivalence class**.

2. $(\text{mod } C)$ tells us what **operation** we applied to A and B.

3. when we have both of these, we call "$\equiv$" **congruence modulo** C.

e.g. $26 \equiv 11 \ (\text{mod } 5)$

$26 \bmod 5 = 1$ so it is in the equivalence class for 1,
$11 \bmod 5 = 1$ so it is in the equivalence class for 1, as well.

1.10 MULTIPLICATIVE INVERSES

The modular inverse of a in the ring of integers modulo m is an integer x such that

$$ax \equiv 1 \pmod{m}.$$

From the Euclidean division algorithm and Bézout's identity, we have the following result about the existence of multiplicative inverses in modular arithmetic:

If a and N are integers such that $\gcd(a, N) = 1$, then there exists an integer x such that $ax \equiv 1 \pmod{N}$.

x is called the **multiplicative inverse** of a modulo N.

SOLVED EXAMPLES

1. **$7^{-1} \bmod 160$**

$160 = 22 \times 7 + 6$ | $6 = 160 - (22 \times 7)$

$7 = 1 \times 6 + 1$ | $1 = 7 - (1 \times 6)$

$1 = 7 - (1 \times 6)$

$= 7 - (1 \times (160 - (22 \times 7))$ for 6 substitute

$= 7 - (1 \times (160 - (22 \times 7))$

$= 1x7 - 1x160 + 22x7$ (Taking 7 as common so (1+22)7)

$= 23\ (7) - 1x160$

$7^{-1} \bmod 160 = 23$

Solve: $23^{-1} \bmod 26$

$26 = 1 \times 23 + 3$ $\quad 3 = 26 - (1 \times 23)$

$23 = 7 \times 3 + 2$ $\quad 2 = 23 - (7 \times 3)$

$3 = 1 \times 2 + 1$ stop $\quad 1 = 3 - (1 \times 2)$

$1 = 3 - (1 \times 2)$

$= 3 - (1 \times (23 - (7 \times 3))$

$= 3 - (1 \times 23) + (7 \times 3)$

$= 1 \times 3 - (1 \times 23) + (7 \times 3)$

$= 8 \times 3 - 1 \times 23$

$= 8 \times (26 - (1 \times 23)) - (1 \times 23))$

$= 26 \times 8 - 9(23)$

$-9 + 26 = 17$ (for negative number add the divisor from the problem)

$23^{-1} \bmod 26 = 17$

1.11 PRIME NUMBERS

Prime numbers are the positive integers having only two factors, 1 and the integer itself

For example,

Factors of 6 are 1,2,3 and 6, which are four factors in total.

But factors of 7 are only 1 and 7, totally two

Hence, 7 is a prime number but 6 is not, instead it is a composite number.

****Always remember that 1 is neither prime nor composite**

Another way of defining Prime Number is - It is a positive number or integer, which is not a product of any other two positive integers.

1.11.1 Relative Prime Numbers

The numbers 'a' & 'b' are said to be Relative Prime numbers if 'a' & 'b' does not have a common factor

i.e., GCD(a, b)= 1

GCD – Greatest Common Divisor

For example,

Assume a=15 & b = 28

Factors of 15 are 1,3,5

Factors of 28 are 1,2,4,7,14

GCD is the largest number that divides both of them.

In this case 1 is the common divisor

So GCD (15,28) = 1 ,Hence 15 & 28 are relatively Prime numbers

Practice Problem: Find GCD of 36 and 60

1.12 EULERS AND FERMATS THEOREM

1.12.1 Euler's Totient Function

Euler's Totient function is also known as PHI function ($\varphi(n)$) or $\phi(n)$

Euler's Totient function for any given number 'n' is defined as the Total count of the numbers which are relatively Prime to 'n' and are less than 'n'.

Example 1:

Assume 'n' = 10

Consider the numbers which are lesser than n(in this case 10).Then the Relative Prime numbers for 10 are 1,3,7,9.

Hence $\varphi(n))$ or $\phi(n) = 4$ (Since Relative Prime for 10 is 1,3,7,9) (Total 4 Numbers).

If the given number 'n' is a Prime number then, $\phi(p) = P-1$

Example 2:

Consider n=7 (Since 7 is a prime number Euler's Totient function $\varphi(7) = 7-1 = 6$.

Example 3: n=13. Find $\varphi(13)$.

$\varphi(13) = ?$

Example 4: n= 11. Find $\varphi(11)$.

Example 5: (non prime number)

n = 14. Find φ(14) = ?

Hint: Count of Relative Prime numbers for 14.

Example 6:

Determine φ(37) and φ(35).

Because 37 is prime, all of the positive integers from 1 through 36 are relatively prime to 37.

Thus **φ(37) = 36.**

To determine φ(35), we list all of the positive integers less than 35 that are relatively prime to it:

1, 2, 3, 4, 6, 8, 9, 11, 12, 13, 16, 17, 18, 19, 22, 23, 24, 26, 27, 29, 31, 32, 33, 34.

There are 24 numbers on the list, so **φ(35) = 24.**

1.12.2 Euler's Theorem

Euler's theorem states that for every a and n that are relatively prime:

$$a^{\phi(n)} \equiv 1 (\mathrm{mod}\ n)$$

$a = 3; n = 10; \phi(10) = 4$	$a^{\phi(n)} = 3^4 = 81 \equiv 1 (\mathrm{mod}\ 10) = 1\ (\mathrm{mod}\ n)$
$a = 2; n = 11; \phi(11) = 10$	$a^{\phi(n)} = 2^{10} = 1024 \equiv 1 (\mathrm{mod}\ 11) = 1\ (\mathrm{mod}\ n)$

aφ(n) ≡ 1 mod n

φ(n) is the totient function is defined as the number of positive integers less than n that are co prime to n. (n>=1)

φ(5) ={ 1 2 3 4}

Proof:

aφ(n) ≡ 1 mod n is true if n is prime, because in that case φ(n) = (n 1) and Fermat's theorem holds. However, it also holds for any integer n. Recall that φ(n) is the number of positive integers less than n that are relatively prime to n. Consider the set of such integers, labeled as follows:

$$R\ \{x_1, x_2, \ldots, x_{\phi(n)}\}$$

That is, each element x_i of R is a unique positive integer less than n with

$\gcd(x_i , n) = 1$. Now multiply each element by a, modulo n:

$$S = \{(ax_1 \bmod n), (ax_2 \bmod n), \ldots, (ax_{\phi(n)} \bmod n)\}$$

The set S is a permutation of R, by the following line of reasoning:

1. Because a is relatively prime to n and x_i is relatively prime to n, ax_i must also be relatively prime to n. Thus, all the members of S are integers that are less than n and that are relatively prime to n. 2. There are no duplicates in S.

. If $ax_i \bmod n = ax_j \bmod n$ then $x_i = x_j$

Alternative form

Euler's Theorem states that , If 'p' & 'q' are two prime numbers such that $p \neq q$ & n =pq , Then $\phi(n) \equiv (p-1)*(q-1)$.

Example 1:

Assume 'p' = 2 & 'q' = 5

n = pq

= 2*5

= 10

So $\phi(10) \equiv (2-1)*(5-1)$

= 4

Example 2:

Consider p=7 & q= 11 , n= 7*11= 77 .Find φ(77).

$\varphi(77) \equiv (7-1)*(11-1) = 6*10 = 60$

1.12.3 Fermat's Theorem

Fermat's theorem states the following: If p is prime and a is a positive integer not divisible by p, then

Proof:

Proof: Consider the set of positive integers less than p:{1,2,..., p 1} and multiply each element by a, modulo p, to get the set X = {a mod p, 2a mod p, . . . (p 1)a mod p}.

None of the elements of X is equal to zero because p does not divide a. Furthermore no two of the integers in X are equal. To see this, assume that

ja ≡ka(mod p) where 1 j < k p 1. Because a is relatively prime[5] to p, we can eliminate a from both sides of the equation ,resulting in:

j≡ k(mode p). This last equality is impossible because j and k are both positive integers less than p. Therefore, we know that the (p 1) elements of X are all positive integers, with no two elements equal. We can conclude the X consists of the set of integers {1,2,..., p 1} in some order. Multiplying the numbers in both sets and taking the result mod p yields

a x 2a x ... x (p 1) ≡ [(1 x 2 x ... x (p 1)](mode p)

ap1(p 1)! ≡ (p 1)!(mod p)

We can cancel the (p 1)! term because it is relatively prime to p

$a = 7, p = 19$
$7^2 = 49 \equiv 11 (\text{mod } 19)$

$7^4 \equiv 121 \equiv 7 (\text{mod } 19)$
$7^8 \equiv 49 \equiv 7 (\text{mod } 19)$
$7^{16} \equiv 121 \equiv 7 (\text{mod } 19)$
$a^{p1} = 7^{18} = 7^{16} \times 7^2 \equiv 7 \times 11 \equiv 1 (\text{mod } 19)$

An alternative form of Fermat's theorem is also useful: If p is prime and a is a positive integer, then

$$a^p \equiv a (\text{mod } p)$$

$p = 5, a = 3$	$a^p = 3^5 = 243 \equiv 3 (\text{mod } 5) = a (\text{mod } p)$
$p = 5, a = 10$	$a^p = 10^5 = 100000 \equiv 10 (\text{mod } 5) = 0 (\text{mod } 5) = a (\text{mod } p)$

Fermat's Theorem/ Fermat's Little Theorem States that if 'P' is a Prime number and 'a' is a Positive Integer which is not Divisible by 'P' then

$$a^{P-1} \equiv 1 \bmod P$$

Example: Let a = 3 and P = 7

$$a^{P-1} \equiv 1 \bmod P$$

$$3^{7-1} \equiv 1 \bmod 7 \equiv 3^6 \bmod 7$$

$$(3^2)^3 \bmod 7 \equiv (9 \bmod 7)^3$$

$$\equiv (2 \bmod 7)^3$$

$$\equiv 2^3 \bmod 7 \equiv 8 \bmod 7 = 1$$

Hence proved.

Key Points

● A prime number is an integer that can only be divided without remainder by positive and negative values of itself and 1. Prime numbers play a critical role both in number theory and in cryptography.

● Two theorems that play important roles in public-key cryptography are Fermat's theorem and Euler's theorem.

● An important requirement in a number of cryptographic algorithms is the ability to choose a large prime number. An area of ongoing research is the development of efficient algorithms for determining if a randomly chosen large integer is a prime number.

Solved Examples

Solve using Fermats Theorem

If n is prime and x is a positive integer not divisible by n then

$$\mathbf{x^{n-1} \equiv 1 \bmod n}$$

n- prime no.
x- is not divisible by n

x and n ---- coprime

Example 1:

x= 3 n=5

$3^{5-1} \equiv 3^4 = 81$

$81 = 1 \bmod 5$

Another form of fermats

$x^n \equiv x \bmod n$

Example 2:

x= 3 n=5

$x^n = 3^5 = 243$

$243 \equiv 3 \bmod 5$

Example 3:

$2^{16} \bmod 17$

By fermats

$x^{n-1} = 1 \bmod n$

$2^{17-1} = 1 \bmod 17$

$2^{16} \bmod 17 = 1$

Example 4:

$7^{61} \bmod 31$

x=7 n=31

$x^{n-1} = \bmod n$

$7^{31-1} = 1 \bmod 31$

$7^{30} \bmod 31 = 1$

Now,

$7^{61} = 7^{(30x2)+1}$

$= (7^{30})^2 . 7^1$

$7^{61} \bmod 31 = (7^{30})^2 . 7^1 \bmod 31$

$[(7^{30})^2 \bmod 31 \times 7^1 \bmod 31] \bmod 31$

$[1 \times 7^1 \bmod 31] \bmod 31$

7

REVIEW QUESTIONS:

1. Prove Fermat's Theorem .Consider a=2, P= 5
   i. $a^{P-1} = 1 \bmod P$
2. Prove Fermat's Theorem Using a=3, P= 7
3. Solve $31^{-1} \bmod 37$
4. Solve $5^{-1} \bmod 96$
5. Solve $16^{-1} \bmod 23$
6. Infer Eulers totient function
7. Compare passive and active attack
8. Interpret availability and authenticity
9. Distinguish threat and attacks.
10. Interpret Confidentiality and Integrity
11. Prove congruence modulo using an example.
12. Mention the types of active attacks.
13. State non repudiation.
14. Determine $\varphi(37)$ and $\varphi(35)$.
15. Explain computer security challenges.
16. Elaborate about security attacks with neat diagrams
17. Discuss the Key security concepts with the TRIAD diagram
18. State and prove the Eulers and Fermats theorem using examples.
19. With a neat sketch explain network security model.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT - 2

SCS1316 - NETWORK SECURITY

UNIT - II CRYPTOGRAPHY BASICS

Syllabus:

Terminologies – Cryptography – Classification: based on operation, number of keys used, Processing - Crypt analysis: Types - Classical Encryption - Substitution Cipher: Ceaser Cipher, Brute Force attack, Vignere Cipher, One time pad, Transposition Cipher: Rail fence Cipher, Simple row column Transfer, Play Fair Cipher, 2X2 Hill cipher - Stream cipher - Block Cipher - Modes of operation – DES – AES - RSA algorithm

2.1 Terminologies of Cryptography:

Cryptography

The many schemes used for encryption constitute the area of study known as cryptography

Crypt analysis

Techniques used for deciphering a message without any knowledge of the enciphering details fall into the area of cryptanalysis. Cryptanalysis is what the layperson calls “breaking the code.”

Cryptology

The areas of cryptography and cryptanalysis together are called cryptology

Cipher

Encryption scheme is known as a cryptographic system or a cipher

Plain Text

This is the original intelligible message or data that is fed into the algorithm as input.

Cipher Text

This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different cipher texts. The cipher text is an apparently random stream of data and, as it stands, is unintelligible.

Secret key

The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the algorithm depend on the key.

Encryption

The process of converting from plaintext to cipher text

Decryption

The process of restoring the plaintext from the cipher text

Enciphering Algorithm

The encryption algorithm performs various substitutions and transformations on the plaintext

Deciphering Algorithm

This is essentially the encryption algorithm run in reverse. It takes the cipher text and the secret key and produces the original plaintext.

Threat
A potential for violation of security which exists when there is a circumstance, capability, action, or event, that could breach security and cause harm. That is, a threat is a possible danger that might exploit vulnerability.

Attack
An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

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. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.

Principles of Security
Symmetric Cipher Model
A symmetric encryption scheme has five ingredients. They are Plain Text, Encryption Algorithm, Secret Key, Decryption Algorithm, Cipher Text
There are two requirements for secure use of conventional encryption:

- We need a strong encryption algorithm. At a minimum, we would like the algorithm to be such that an opponent who knows the algorithm and has access to one or more cipher texts would be unable to decipher the cipher text or figure out the key.
- Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

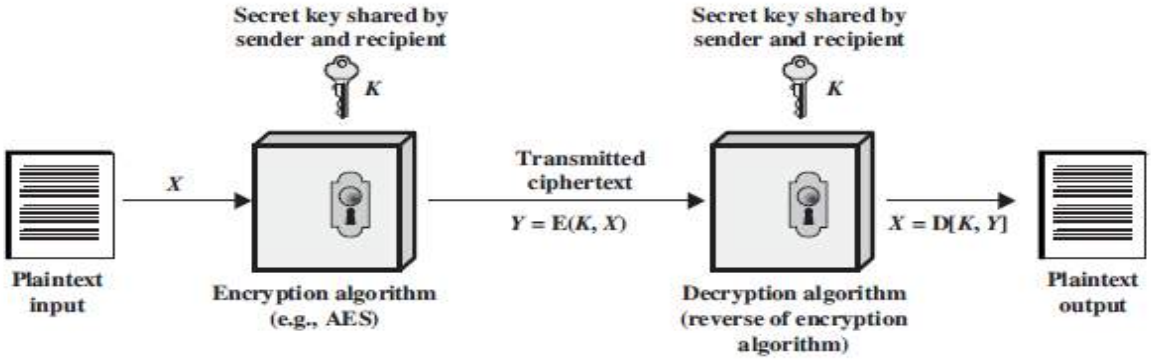


Fig. 2.1 Model of Symmetric Encryption

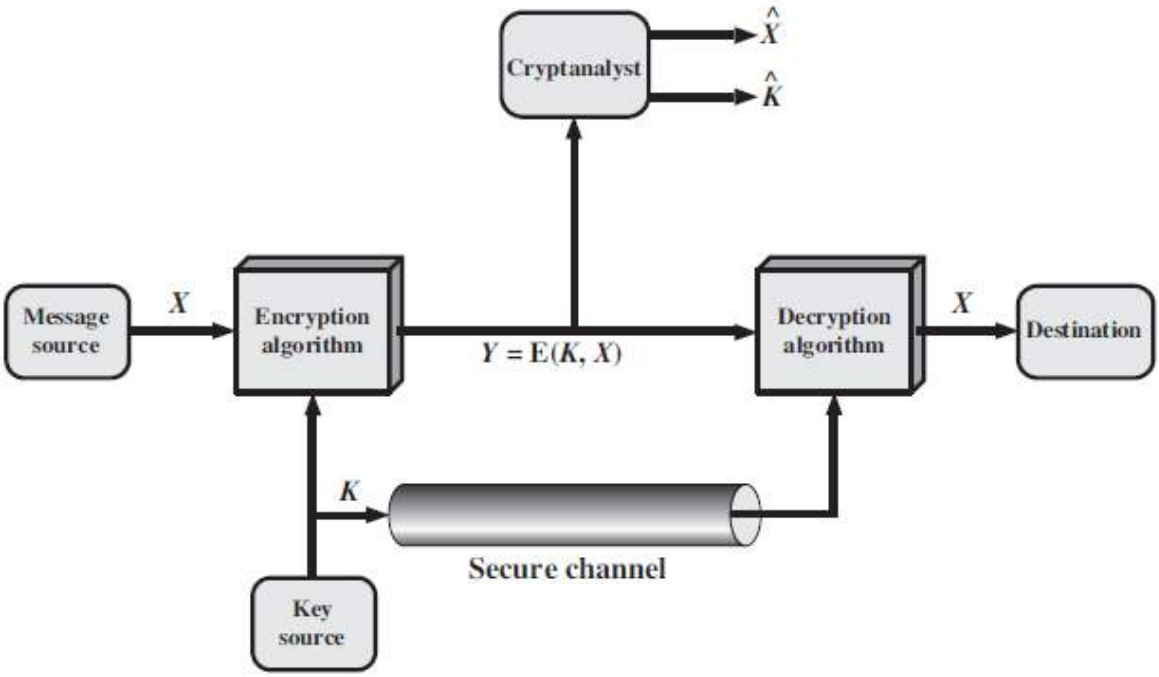


Fig. 2.2 Model of Symmetric Cryptosystem

Confidentiality: This term covers two related concepts:

- Data confidentiality: Assures that private or confidential information is not made available or disclosed to unauthorized individuals.
- Privacy: Assures that individuals control or influence what information related to them may be collected and stored and by whom and to whom that information may be disclosed.

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

- Peer Entity Authentication: Used in association with a logical connection to provide confidence in the identity of the entities connected.
- Data-Origin Authentication: In a connectionless transfer, provides assurance that the source of received data is as claimed.

Integrity: This term covers two related concepts:

- Data integrity: Assures that information and programs are changed only in a specified and authorized manner.
- System integrity: Assures that a system performs its intended function in an unimpaired manner, free from deliberate or inadvertent unauthorized manipulation of the system.

Non-repudiation

Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.

- Nonrepudiation, Origin: Proof that the message was sent by the specified party

- Nonrepudiation, Destination: Proof that the message was received by the specified party

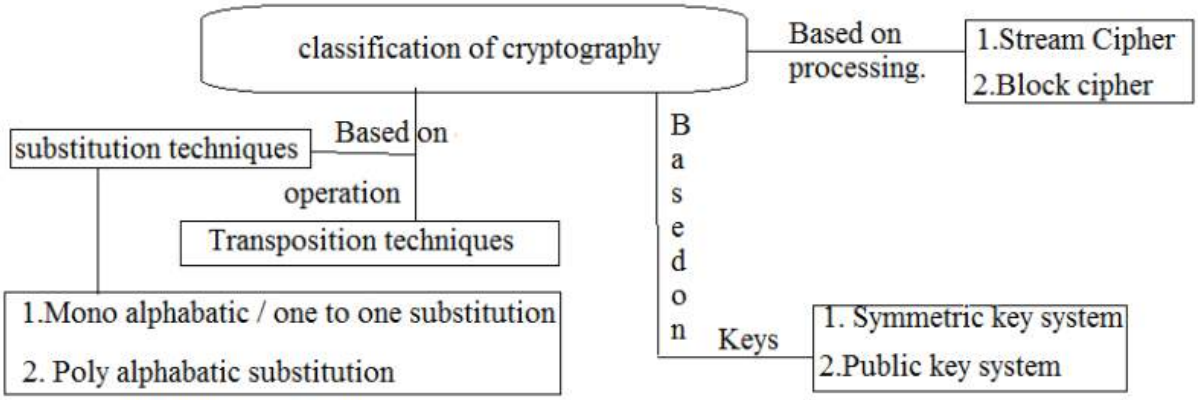
Access Control

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

Availability

Assures that systems work promptly and service is not denied to authorized users.

2.2 Cryptography Classifications:



2.3 Types of cryptanalysis: Cryptanalysis is the science of recovering the plaintext of a message without access to the key. Successful cryptanalysis may recover the plaintext or the key. The two basic categories of cryptanalysis are 1. Linear Cryptanalysis and 2. Differential cryptanalysis

Linear Cryptanalysis: Linear cryptanalysis is a known plaintext attack, in which the attacker studies probabilistic linear relations known as linear approximations between parity bits of the plaintext, the Ciphertext and the secrete key. In this technique, the attacker obtains high probability approximations for the parity bit of the secrete key by analysing the parity bits of the known plaintexts and cipher texts. By use of several techniques such as the auxiliary technique, the attacker can extend the attack to find more bits of the secret key.

Differential Cryptanalysis: Differential cryptanalysis can be described as a general form of cryptanalysis that is primarily applicable to block ciphers, cryptographic hash functions. In other words, it entails a careful analysis of how differences in information input can affect the resulting difference at the output. In block cipher, differential analysis can be described as a set of techniques for tracing differences through the network of transformation, discovering where the cipher exhibits what is known as non-random behaviour and exploiting such details to recover the secrete key (cryptography key). In the process, observing the desired output difference between the two chosen or unknown plaintext inputs suggests possible key values.

2.4 Classical Encryption Techniques

2.4.1 Substitution Cipher

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

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

For example,
Plain text : meet me after the toga party
Cipher Text : PHHW PH DIWHU WKH WRJD SDUWB

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

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

Let us assign a numerical equivalent to each letter:

A	**b**	**c**	**d**	**e**	**F**	**g**	**h**	**i**	**j**	**k**	**l**	**m**
0	1	2	3	4	5	6	7	8	9	10	11	12
N	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
13	14	15	16	17	18	19	20	21	22	23	24	25

Then the algorithm can be expressed as follows.
For each plaintext letter p, substitute the cipher text letter C
$C = E(3, p) = (p + 3) \bmod 26$

A shift may be of any amount, so that the general Caesar algorithm is
$C = E(k, p) = (p + k) \bmod 26$
where k takes on a value in the range 1 to 25.

The decryption algorithm is simply
$p = D(k, C) = (C\ k) \bmod 26$
If it is known that a given cipher text is a Caesar cipher, then a brute-force cryptanalysis is easily performed: Simply try all the 25 possible keys.

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

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

Play Fair Cipher

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

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

In this case, the keyword is **monarchy**.

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

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

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

Hill cipher

Another interesting multi-letter cipher is the Hill cipher, developed by the mathematician Lester Hill in 1929. The encryption algorithm takes m successive plaintext letters and substitutes for them m cipher text letters.

The substitution is determined by m linear equations in which each character is assigned a numerical value (a = 0, b = 1 ... z = 25).
For m = 3, the system can be described as follows:

$c_1 = (k_{11}P_1 + k_{12}P_2 + k_{13}P_3) \bmod 26$
$c_2 = (k_{21}P_1 + k_{22}P_2 + k_{23}P_3) \bmod 26$
$c_3 = (k_{31}P_1 + k_{32}P_2 + k_{33}P_3) \bmod 26$

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

$$\begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} P_1 & P_2 & P_3 \end{pmatrix} \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \text{mod} 26$$

Or

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

$P = D(K, C) = CK^{-1} \text{ mod } 26 = PKK^{-1} = P$

One Time Pad

An Army Signal Corp officer, Joseph Mauborgne suggested using a **random key** that is as **long as the message**, so that the key need not be repeated. In addition, the key is to be used to encrypt and decrypt a single message, and then is discarded. Each new message requires a new key of the same length as the new message. Such a scheme, known as a one-time pad, is **unbreakable**.

It produces random output that bears no statistical relationship to the plaintext. Because the cipher text contains no information whatsoever about the plaintext, there is simply no way to break the code.

An example should illustrate our point. Suppose that we are using a 27 characters in which the twenty-seventh character is the space character, but with a one-time key that is as long as the message.

Consider the
cipher text : ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS
We now show two different decryptions using two different keys:

key 1: pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih
plain text: mr mustard with the candlestick in the hall

key 2: mfugpmiydgaxgoufhkllmhsqdqogtewbqfgyovuhwt
plain text: miss scarlet with the knife in the library

If the actual key were produced in a truly random fashion, then the cryptanalyst cannot say that one of these two keys is more likely than the other. Thus, there is no way to decide which key is correct and therefore which plaintext is correct. Therefore, the code is unbreakable.

2.4.2 Transposition Cipher

A kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

Rail Fence Technique

The simplest transposition cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

For example,
to encipher the message "**meet me after the toga party**" with a rail fence of depth 2, we write the following:

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

The encrypted message is **"MEMATRHTGPRYETEFETEOAAT"**

Simple Columnar Technique

A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm.
For example,
Key: 3 4 2 1 5 6 7
Plaintext: **a t t a c k p o s t p o n e d u n t i l t w o a m x y z**

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{pmatrix} a & t & t & a & c & k & p \\ o & s & t & p & o & n & e \\ d & u & n & t & i & l & t \\ w & o & a & m & x & y & z \end{pmatrix} \end{array}$$

Cipher text: **TTNAAPTMTSUOAODWCOIXKNLYPETZ**

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext.

For the type of columnar transposition just shown, cryptanalysis is fairly straightforward and involves laying out the cipher text in a matrix and playing around with column positions. Digram and trigram frequency tables can be useful.

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

Key: 3 4 2 1 5 6 7

1 2 3 4 5 6 7

$$\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{pmatrix} t & t & n & a & p & t & m \\ t & t & s & u & o & a & o \\ d & w & c & o & i & x & k \\ n & l & y & p & e & t & z \end{pmatrix}$$

Cipher text: **NSCYAUOPTTWLTTDNPOIETAXTMOKZ**

2.5 Comparison of Stream Ciphers and Block Ciphers

A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher.

In the ideal case, a one-time pad version of the Vernam cipher would be used, in which the keystream is as long as the plaintext bit stream. 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. This introduces insurmountable logistical problems if the intended data traffic is very large.
Accordingly, for practical reasons, the bit-stream generator must be implemented as an algorithmic procedure, so that the cryptographic bit stream can be produced by both users. In this approach, the bit-stream generator is a key-controlled algorithm and must produce a bit stream that is cryptographically strong. Now, the two users need only share the generating key, and each can produce the keystream.

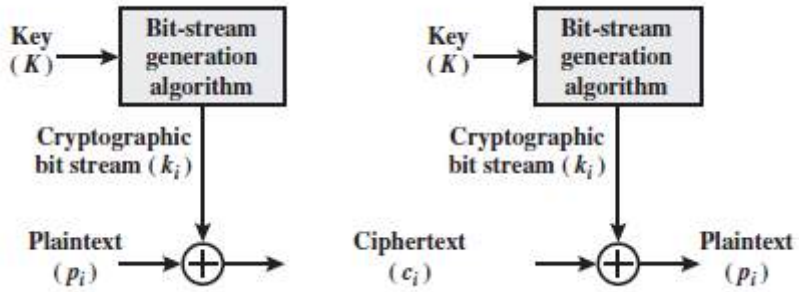


Fig.2.1 Stream Cipher using algorithmic bit-stream generator

A **block cipher** is an encryption/decryption scheme in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically, a block size of 64 or 128 bits is used. In general, they seem applicable to a broader range of applications than stream ciphers. The vast majority of network-based symmetric cryptographic applications make use of block ciphers.

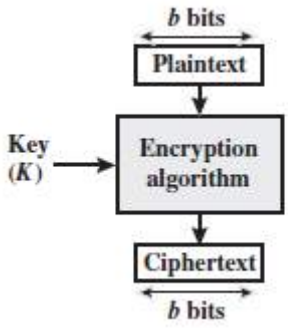


Fig.2.2 Block Cipher

Feistel Block Cipher

Feistel proposed that we can approximate the ideal block cipher by utilizing the concept of a product cipher, which is the execution of two or more simple ciphers in sequence in such a way that the final result or product is cryptographically stronger than any of the component ciphers The essence of the approach is to develop a block cipher with a key length of *k* bits and a block length of bits, allowing a total of 987possible transformations, rather than the ! transformations available with the ideal block cipher.

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

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

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

Diffusion: A cryptographic technique that seeks to obscure the statistical structure of the plaintext by spreading out the influence of each individual plaintext digit over many cipher text digits.

Confusion: A cryptographic technique that seeks to make the relationship between the statistics of the cipher text and the value of the encryption key as complex as possible. This is achieved by the use of a complex scrambling algorithm that depends on the key and the input.

Figure 2.3 depicts the structure proposed by Feistel. The inputs to the encryption algorithm are a plaintext block of length **2w** bits and a key.The plaintext block is divided into two halves L_0 and R_0. The two halves of the data pass through rounds of processing and then combine to produce the ciphertext block. Each round i has as inputs L_{i-1} and R_{i-1} derived from the previous round, as well as a subkey K_i derived from the overall K. In general, the subkeys K_i are different from K and from each other. In Figure 2.3, 16 rounds are used, although any number of rounds could be implemented.

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 K_i . Another way to express this is to say that F is a function of right-half block of w bits and a

subkey of y bits, which produces an output value of length *w* bits: *F*(*REi*, *Ki*+1). Following this substitution, a

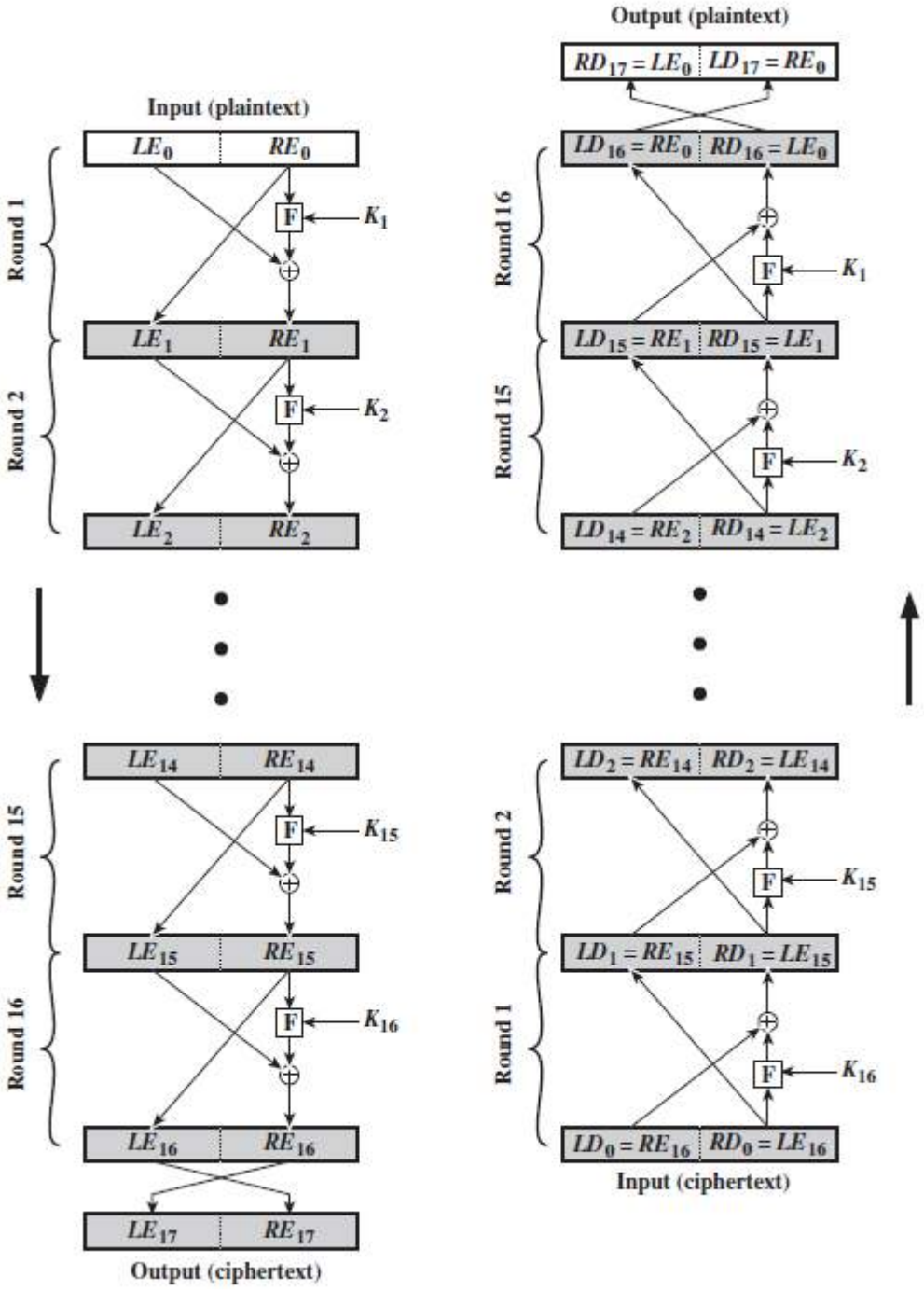


Fig.2.3 Feistel Encryption and Decryption

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

trade off 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 F:** 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.

2.5.1 Block Cipher Modes of Operation

When multiple blocks of plaintext are encrypted using the same key, a number of security issues arise. 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.

(i) Electronic Code Book (ECB) Mode

This mode is a most straightforward way of processing a series of sequentially listed message blocks.

Operation

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

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

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

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

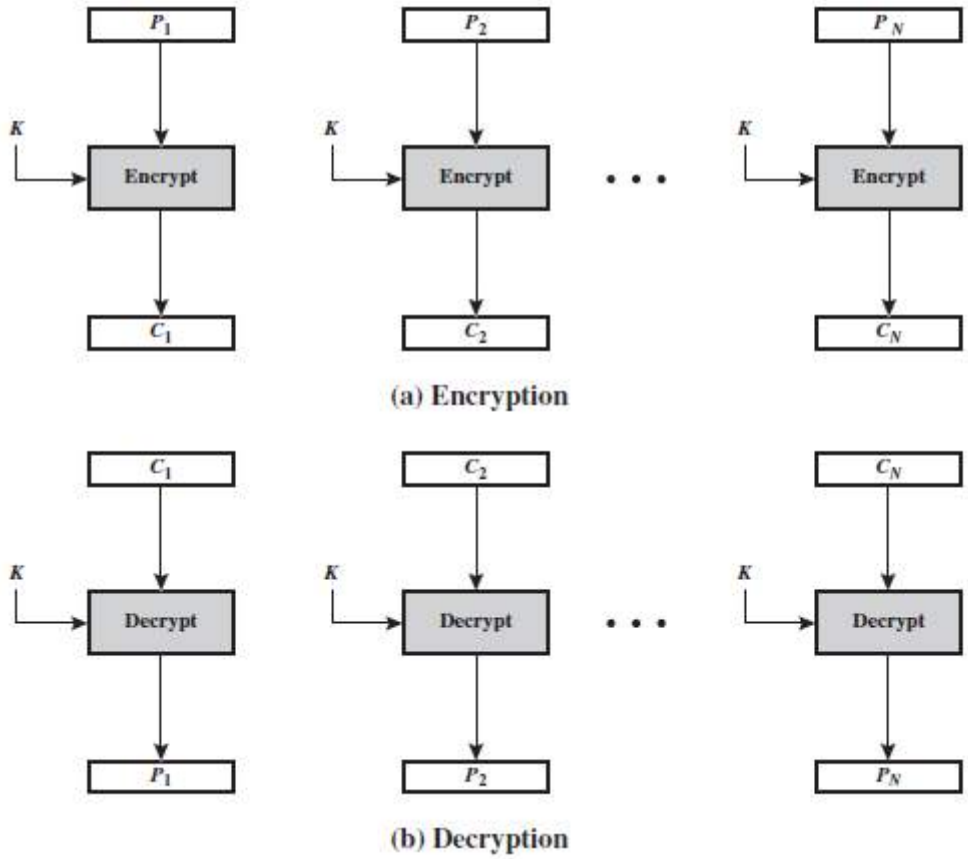


Fig.2.4 ECB - Encryption and Decryption

(ii) Cipher Block Chaining (CBC) Mode

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

Operation

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

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

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

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

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

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

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

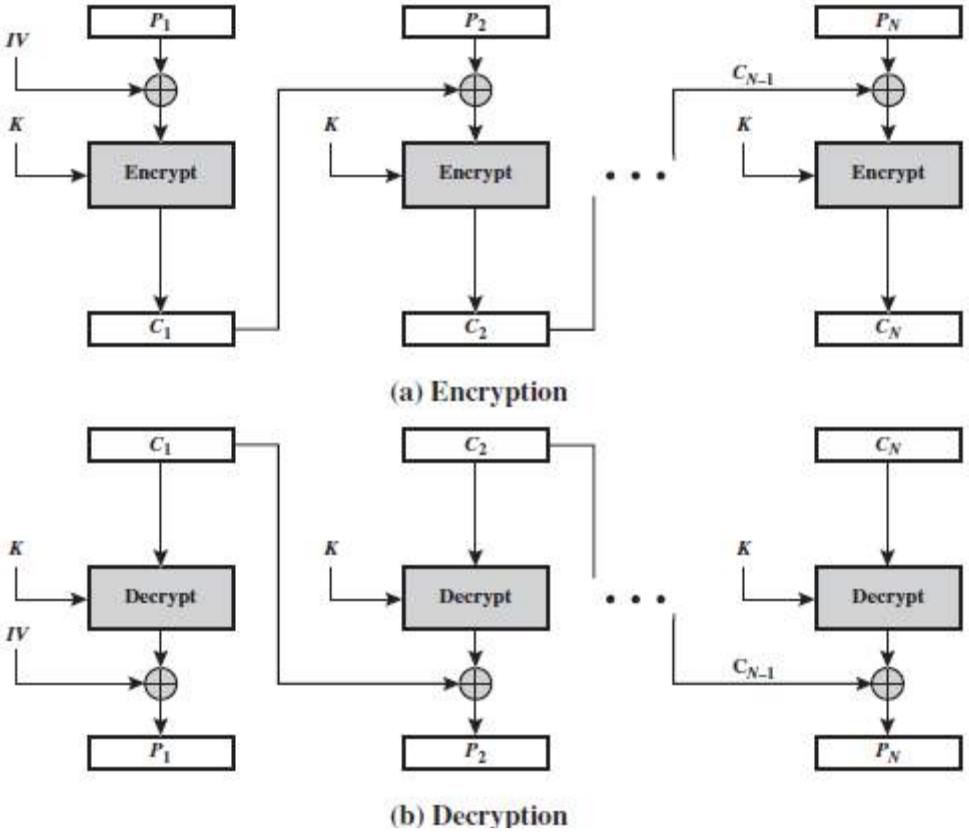


Fig.2.5 CBC - Encryption and Decryption

(iii) Cipher Feedback (CFB) Mode

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

Operation

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

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

$$C_1 = P_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$

$$P_1 = C_1 \oplus \text{MSB}_s[\text{E}(K, \text{IV})]$$

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

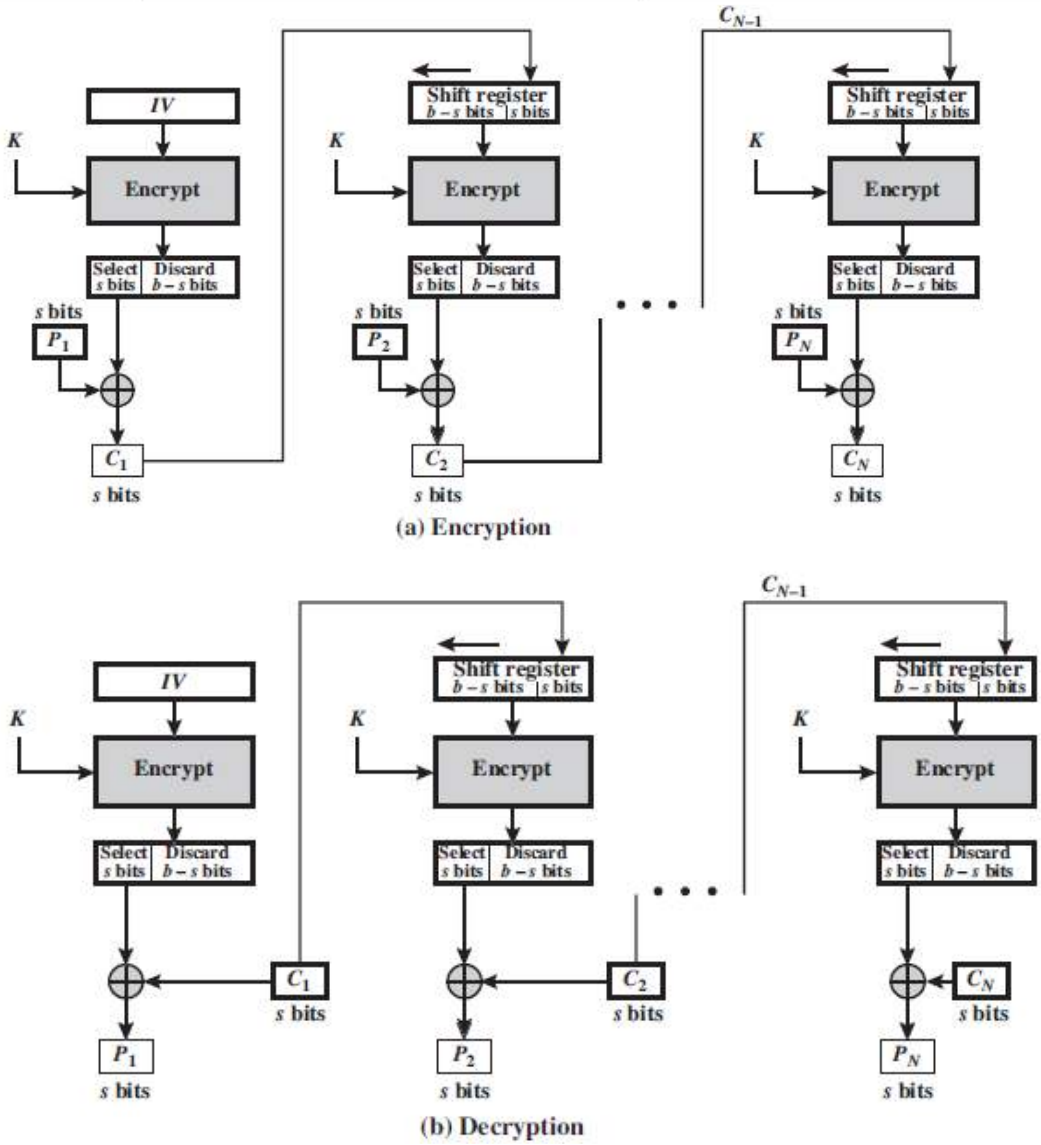


Fig.2.6 CFB - Encryption and Decryption

(iv) Output Feedback (OFB) Mode

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

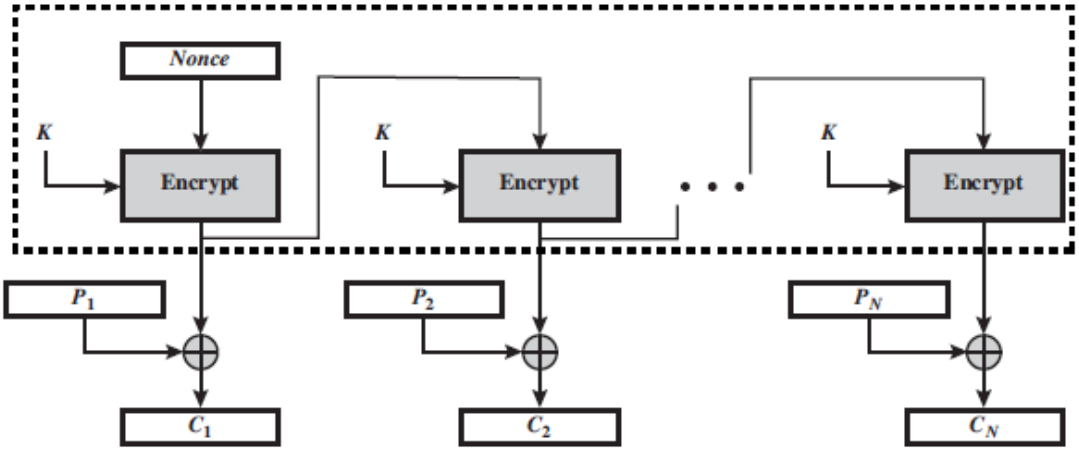
The key stream generated is XOR-ed with the plaintext blocks. The OFB mode requires an IV as the initial random n-bit input block. The IV need not be secret.

The operation is depicted in the following illustration:

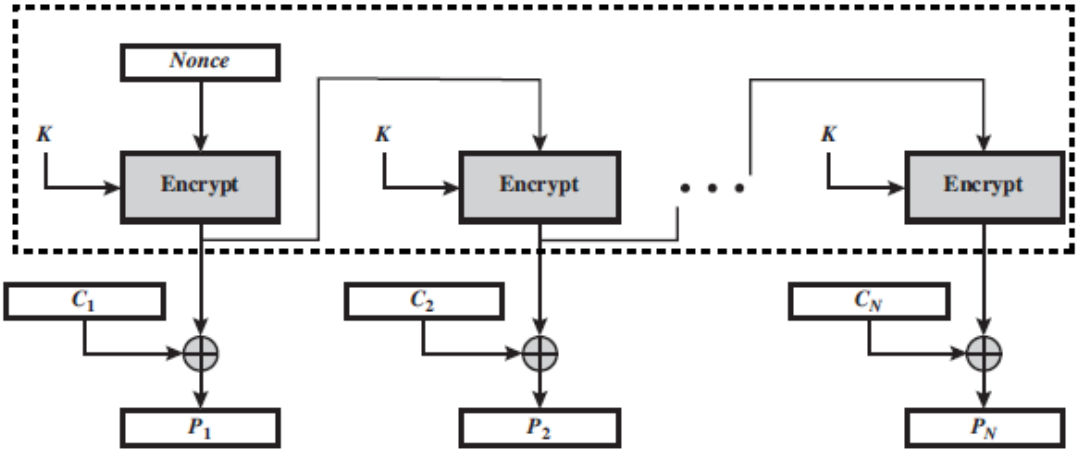
$$C_j = P_j \oplus \text{E}(K, [C_{j-i} \oplus P_{j-1}])$$

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

OFB	$I_1 = Nonce$ $I_j = O_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $C_j = P_j \oplus O_j \quad j = 1, \ldots, N-1$ $C_N^* = P_N^* \oplus \mathrm{MSB}_u(O_N)$	$I_1 = Nonce$ $I_j = \mathrm{LSB}_{b-s}(I_{j-1}) \| C_{j-1} \quad j = 2, \ldots, N$ $O_j = \mathrm{E}(K, I_j) \quad j = 1, \ldots, N$ $P_j = C_j \oplus O_j \quad j = 1, \ldots, N-1$ $P_N^* = C_N^* \oplus \mathrm{MSB}_u(O_N)$



(a) Encryption



(b) Decryption

Fig.2.7 OFB - Encryption and Decryption

(v) Counter (CTR) Mode

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

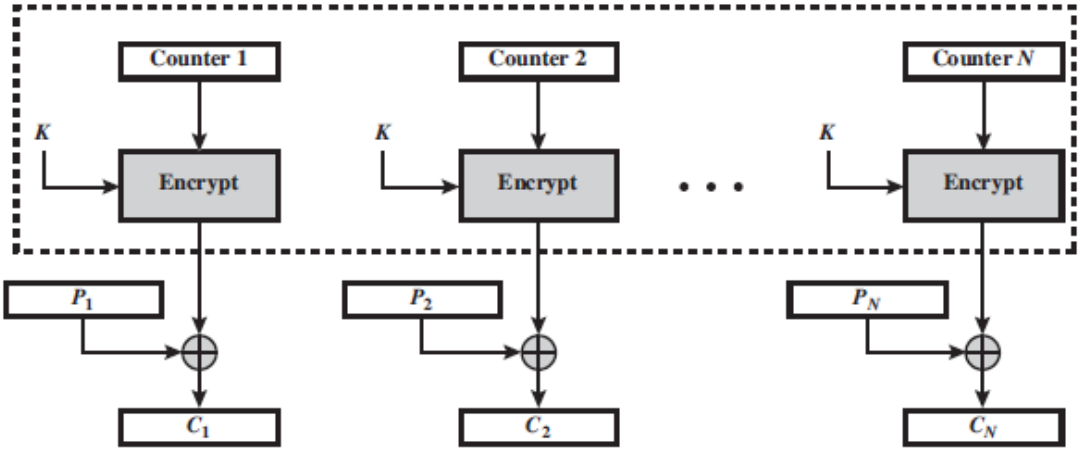
Operation

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

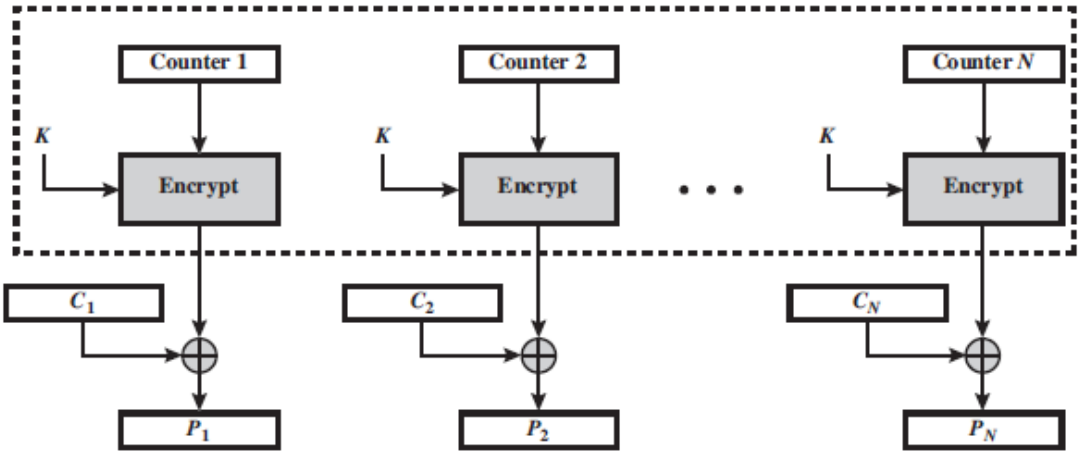
- Load the initial counter value in the top register is the same for both the sender and the receiver. It plays the same role as the IV in CFB (and CBC) mode
- Encrypt the contents of the counter with the key and place the result in the bottom register

- Take the first plaintext block P1 and XOR this to the contents of the bottom register
- The result of this is C1. Send C1 to the receiver and update the counter. The counter
- update replaces the cipher text feedback in CFB mode
- Continue in this manner until the last plaintext block has been encrypted.
- The decryption is the reverse process. The cipher text block is XORed with the output of encrypted contents of counter value. After decryption of each cipher text block counter is updated as in case of encryption

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



(a) Encryption



(b) Decryption

Fig.2.8 CTR - Encryption and Decryption

Table 2.1: Block Cipher Modes of Operation

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

2.6 Data Encryption Standard

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

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

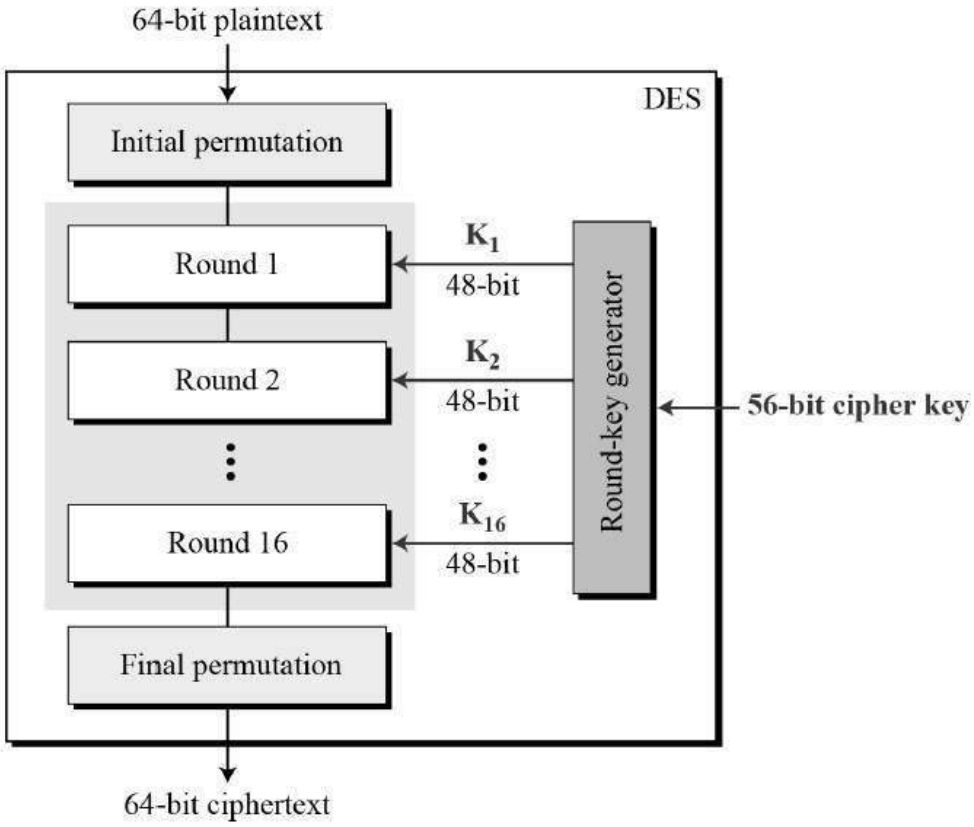


Fig.2.9 DES Structure

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

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

Initial and Final Permutation

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

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

Fig.2.10 Initial Permutation

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

Fig.2.11 Final Permutation

Details of one round in DES

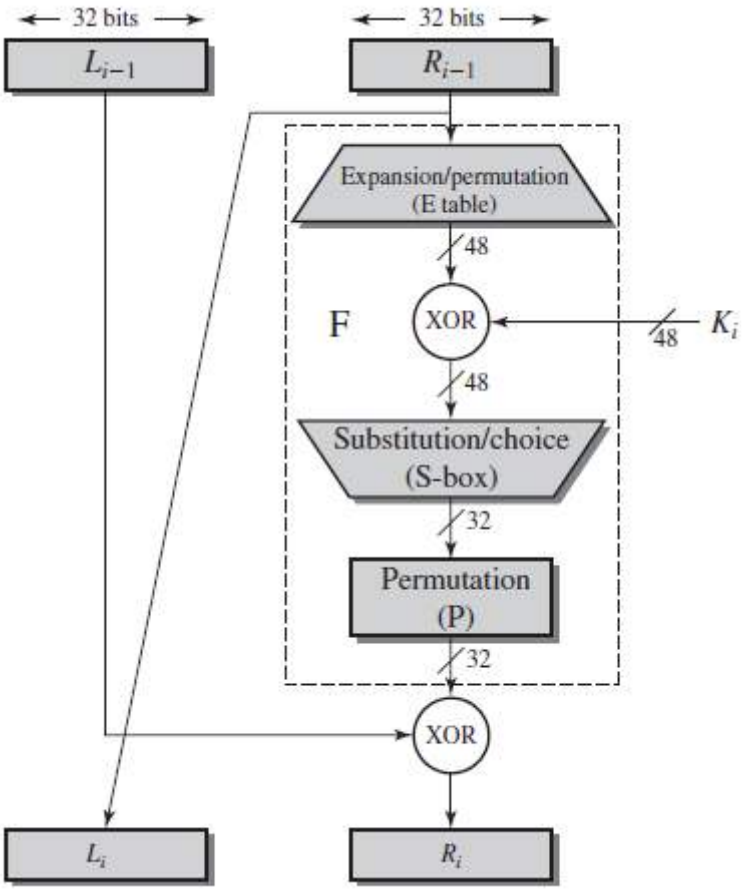


Fig.2.12 One Round in DES

Round Function (F):

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

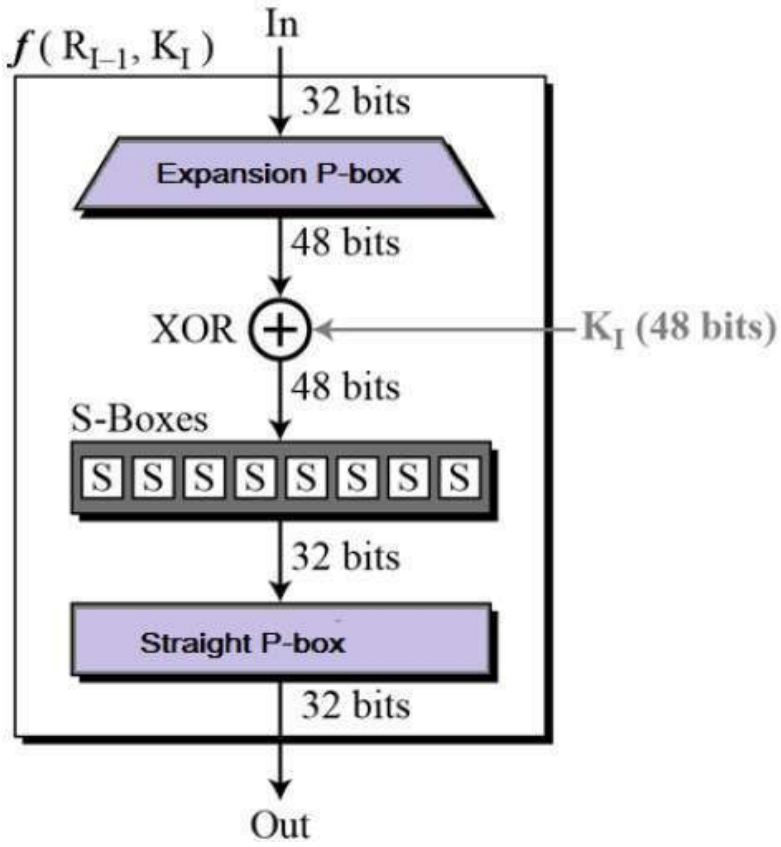


Fig.2.13 Round function

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

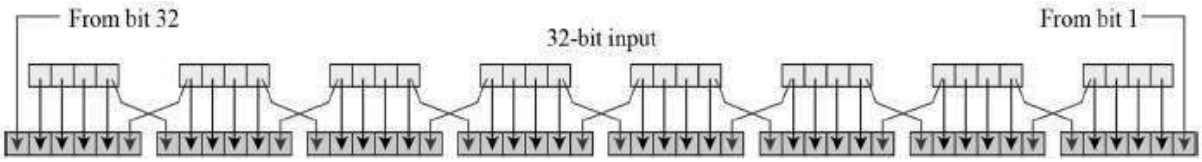


Fig.2.14 Expansion Permutation

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

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

Fig.2.15 Expansion Permutation Table

- **XOR (Whitener).** − After the expansion permutation, DES does XOR operation on the expanded right section and the round key. The round key is used only in this operation.

- **Substitution Boxes.** − The S-boxes carry out the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output. Refer the following illustration

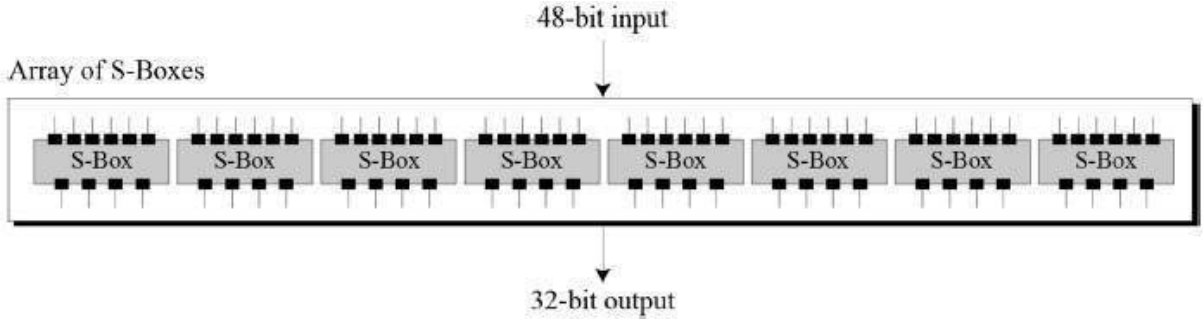


Fig.2.16 Substitution

The S-box rule is illustrated below

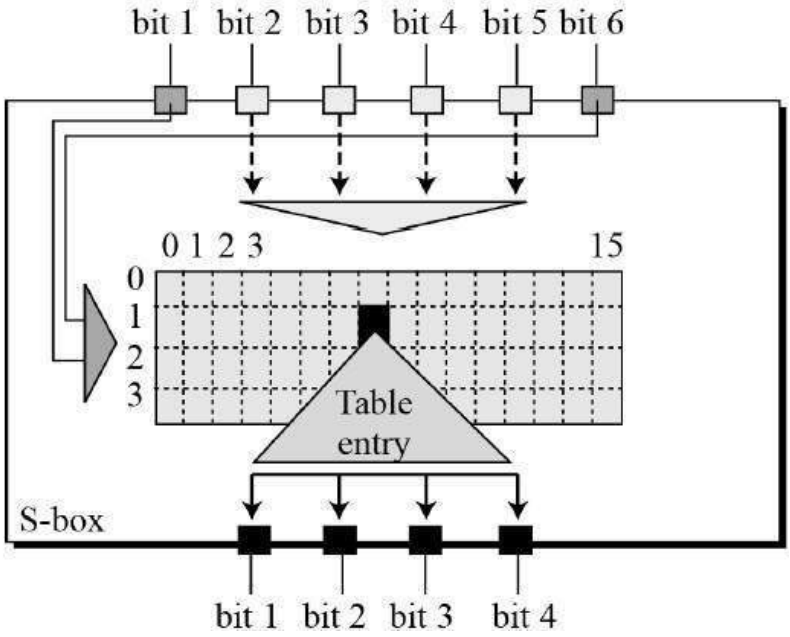


Fig.2.17 S-box rule

- There are a total of eight S-box tables. The output of all eight s-boxes is then combined in to 32-bit section.

S_1

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

S_2

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

S_3

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

S_4

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

S_5

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

S_6

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

S_7

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

S_8

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

Fig.2.18 S-boxes

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

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

Fig.2.19 Permutation Table

Key Generation

The round-key generator creates sixteen 48-bit keys out of a 56-bit cipher key. The process of key generation is depicted in the following illustration

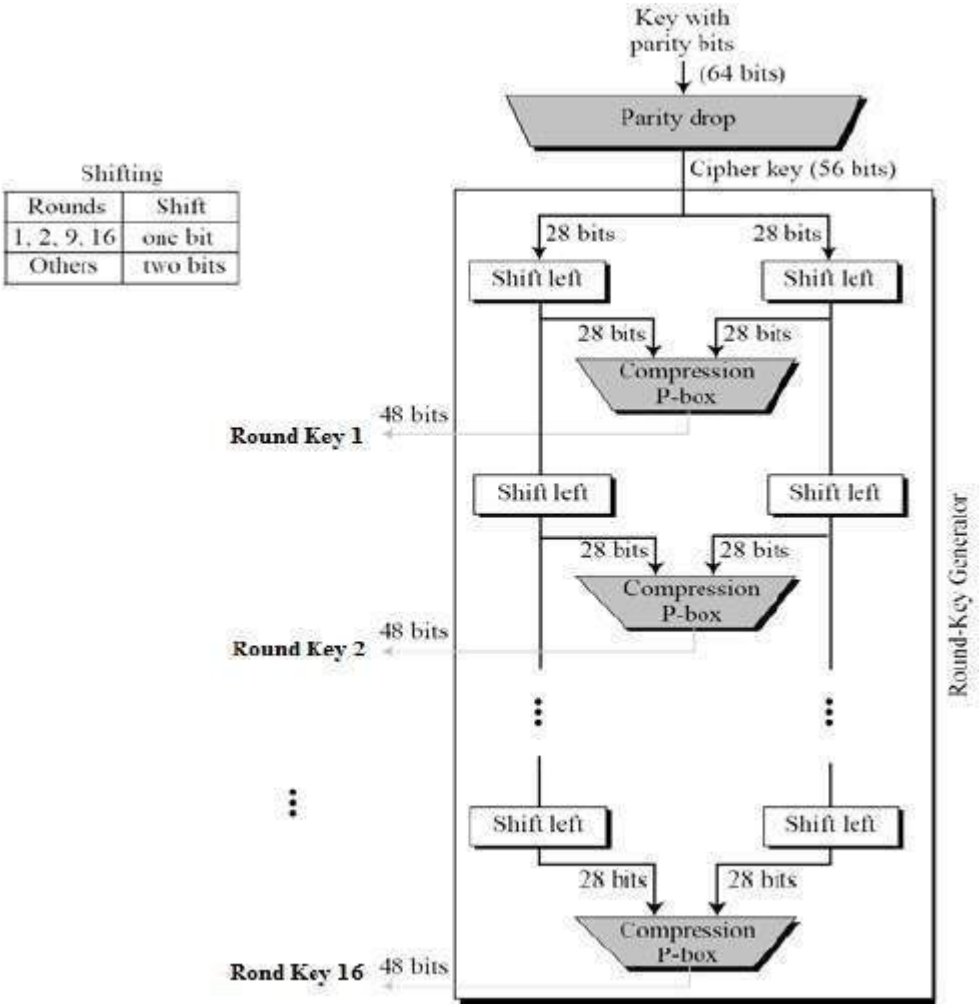


Fig.2.20 Key Generation

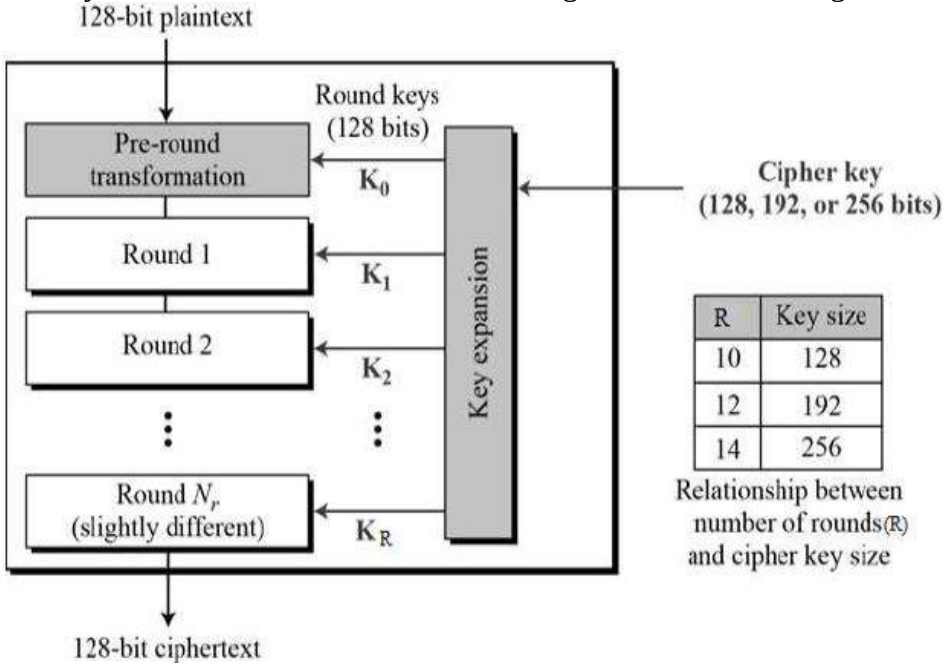
2.7 Advanced Encryption Standard

The more popular and widely adopted symmetric encryption algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.
A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.
The features of AES are as follows:

- Symmetric key symmetric block cipher
- 128-bit data, 128/192/256-bit keys
- Stronger and faster than Triple-DES
- Provide full specification and design details
- Software implementable in C and Java

Operation of AES

AES is an iterative rather than Feistel cipher. It is based on 'substitution–permutation network. It comprises of a series of linked operations, some of which involve replacing inputs by specific outputs (substitutions) and others involve shuffling bits around (permutations). Interestingly, AES performs all its computations on bytes rather than bits. Hence, AES treats the 128 bits of a plaintext block as 16 bytes. These 16 bytes are arranged in four columns and four rows for processing as a matrix: Unlike DES, the number of rounds in AES is variable and depends on the length of the key.
AES uses 10 rounds for 128-bit keys, 12 rounds for 192-bit keys and 14 rounds for 256-bit keys. Each of these rounds uses a different 128-bit round key, which is calculated from the original AES key. The schematic of AES structure is given in the following illustration:



R	Key size
10	128
12	192
14	256

Relationship between number of rounds(R) and cipher key size

Fig.2.21 AES Structure

Encryption Process

Here, we restrict to description of a typical round of AES encryption. Each round comprises of four sub-processes. The first-round process is depicted below:

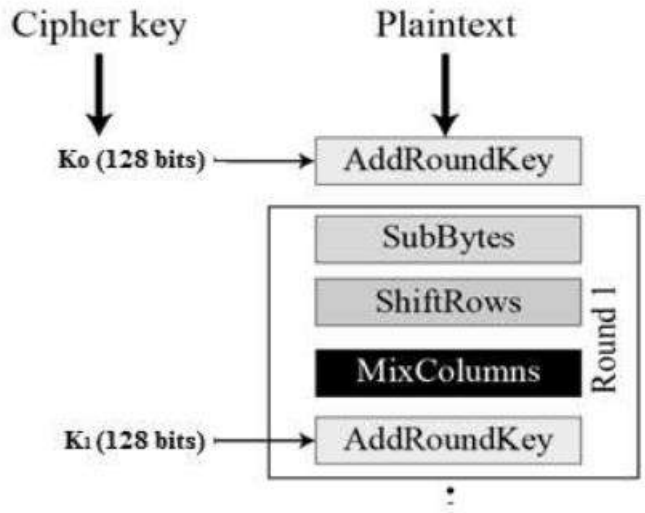


Fig.2.21 One round in AES

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

Shiftrows
Each of the four rows of the matrix is shifted to the left. Any entries that 'fall off' are reinserted on the right side of row. Shift is carried out as follows:

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

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

Addroundkey
The 16 bytes of the matrix are now considered as 128 bits and are XORed to the 128 bits of the round key. If this is the last round then the output is the ciphertext. Otherwise, the resulting 128 bits are interpreted as 16 bytes and we begin another similar round.

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

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

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

2.7 RSA Algorithm:

Diffie and Hellman challenged cryptologists to come up with a cryptographic algorithm that met the requirements for public-key systems.
One of the first successful responses to the challenge was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at MIT and first published in 1978. The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as 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. A typical size for n is 1024 bits, or 309 decimal digits. That is, n is less than 2^{1024}. We examine RSA in this section in some detail, beginning with an explanation of the algorithm. Then we examine some of the computational and cryptanalytical implications of RSA

Description of the Algorithm
RSA makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is, the block size must be less than or equal to $\log_2(n) + 1$; in practice, the block size is i bits, where $2^i < n \leq 2^{i+1}$. Encryption and decryption are of the following form, for some plaintext block M and ciphertext block C.

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

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

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

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

We need to find a relationship of the form
$M^{ed} \bmod n = M$

The preceding relationship holds if e and d are multiplicative inverses modulo $\varphi(n)$, where $\varphi(n)$ is the Euler totient function.

For p,q prime, $\varphi(pq) = (p - 1)(q - 1)$. The relationship between e and d can be expressed as
$$\boldsymbol{ed \bmod \varphi(n) = 1}$$

This is equivalent to saying

$$ed \equiv 1 \bmod \phi(n)$$

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

That is, e and d are multiplicative inverses mod $\varphi(n)$. Note that, according to the rules of modular arithmetic, this is true only if d (and therefore e) is relatively prime to $\varphi(n)$. Equivalently, gcd($\varphi(n)$, d) = 1.

The ingredients are the following:

p, q, two prime numbers	(private, chosen)
$n = pq$	(public, calculated)
e, with $\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$	(public, chosen)
$d \equiv e^{-1} \pmod{\phi(n)}$	(private, calculated)

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

Table 2.2 : RSA algorithm

Key Generation Alice	
Select p, q	p and q both prime, $p \neq q$
Calculate $n = p \times q$	
Calcuate $\phi(n) = (p-1)(q-1)$	
Select integer e	$\gcd(\phi(n), e) = 1; 1 < e < \phi(n)$
Calculate d	$d \equiv e^{-1} \pmod{\phi(n)}$
Public key	$PU = \{e, n\}$
Private key	$PR = \{d, n\}$

Encryption by Bob with Alice's Public Key	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption by Alice with Alice's Public Key	
Ciphertext:	C
Plaintext:	$M = C^d \bmod n$

Example of RSA algorithm

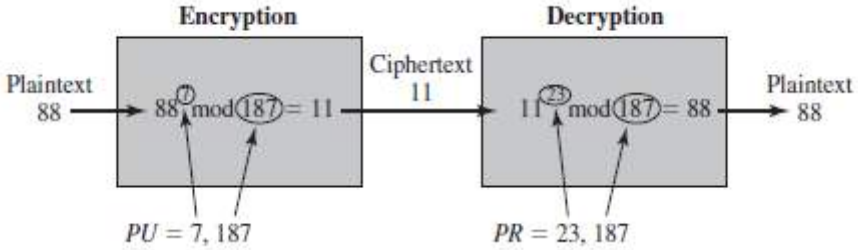


Fig.2.22 Example

Review Questions:

Part-A:

1. What are the essential ingredients of a symmetric cipher?
2. Briefly define the Caesar cipher.
3. What is meant by Brute force attack?
4. Briefly define the multi-letter encryption technique. Give examples.
5. Briefly define the Playfair cipher.
6. What are two problems with the one-time pad?
7. What is a transposition cipher?
8. Why is it important to study the Feistel cipher?
9. What is the difference between a block cipher and a stream cipher?
10. What is the difference between diffusion and confusion?
11. Which parameters and design choices determine the actual algorithm of a Feistel cipher?
12. What is the purpose of the S-boxes in DES?
13. Briefly describe SubBytes.
14. Briefly describe ShiftRows.
15. Briefly describe MixColumns.
16. Why do some block cipher modes of operation only use encryption while others use both encryption and decryption?

Part-B

1. Construct a Play Fair matrix with the key largest. Encrypt this message “Must see you after coming’.
2. Encrypt the message “meet me at the usual place at ten rather than eight” using the Hill cipher with the key. Show your calculations and the result. Show the calculations for the corresponding decryption of the cipher text to recover the original plaintext.
3. Explain security attacks in detail.
4. Discuss about symmetric cryptosystem.
5. Draw Feistel cipher structure and explain.
6. Compare and explain different modes of block cipher operation.
7. Discuss in detail about one round in DES.
8. How 16 Keys are generated in DES? Explain the process in detail.
9. Explain AES in detail
10. Explain RSA algorithm with an example.

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT - 3

SCS1316 - NETWORK SECURITY

UNIT- III SECURITY FUNCTIONS AND DATA SECURITY

Syllabus:

Public Key Crypto system- Diffie-Helmann Key Exchange-Key management Techniques-Hash Functions- Requirements-Hash Algorithm-MD5,SHA_1-Message Authentication Code (MAC)- HMAC-Digital Signature-User Authentication-Kerbroes-X.509 Certificates,X.509 Formats, Public Key Infrastructure- PKIX Architecture-Model-Management Functions.

3.1: PUBLIC KEY CRYPTO SYSTEM:

The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption. The first problem is that of key distribution, and the second one related to digital signature. Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic.

It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key. A public-key encryption scheme has six ingredients Plaintext: This is the readable message or data that is fed into the algorithm as input.

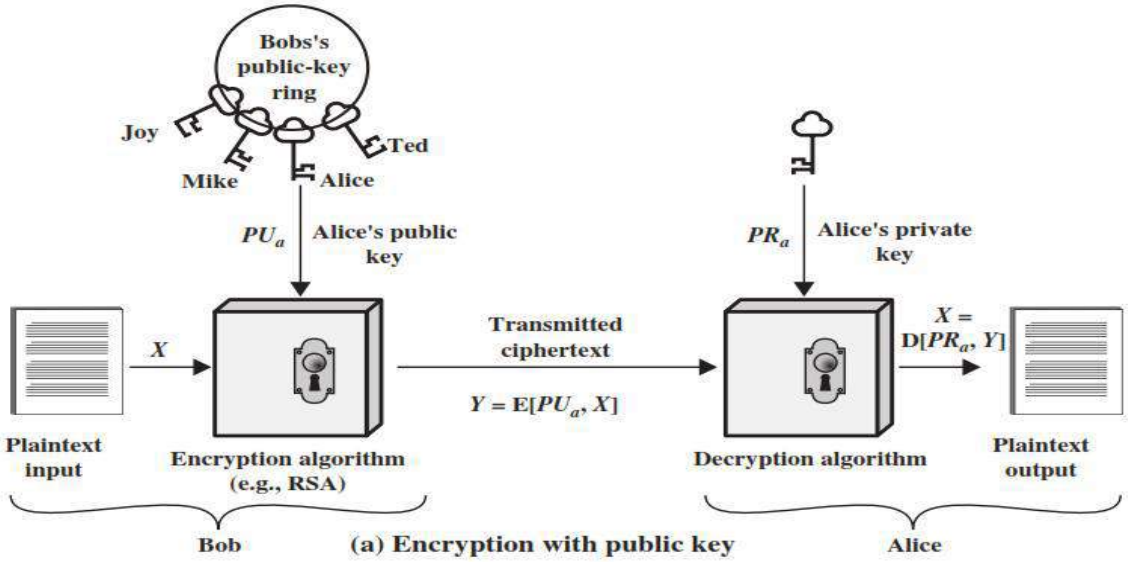


Fig 3.1: Encryption with Public key

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.

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.

2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 3.1, suggests, each user maintains a collection of public keys obtained from others.

3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.

4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key. With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed.

As long as a user's private key remains protected and secret, incoming communication is secure. At any time, a system can change its private key and publish the companion public key to replace its old public key.

3.2 : DIFFIE-HEILMAN KEY EXCHANGE/AGREEMENT ALGORITHM:

Whitefield Diffie and Martin Hellman devised an amazing solution to the problem of key agreement, or key exchange in 1976. This solution is called as the Diffie-Hellman Key Exchange /Agreement Algorithm. The beauty of this scheme is that the two parties, who want to communicate securely, can agree-on a symmetric key using this technique.

It might come as a surprise, but Kl is actually equal to K2! This means that Kl =K2 = K is the symmetric key, which Alice and Bob must keep secret and can henceforth

usefor encrypting/decrypting their messages with. The mathematics behind this is quite interesting.

Let us try to understand what this actually means, in simple terms. (a) Firstly, take a look at what Alice does in step 6. Here, Alice computes: K1 = BX mod n. What is B? From step 4, we have: B = gYmod n. Therefore, if we substitute this value of B in step 6, we will have the following equation: Kl = (g)yr mod n =(g)YXmod n. (b) Now, take a look at what Bob does in step 7. Here, Bob computes: K2 = AY mod n. What is A? From step 2, we have: A = gX mod n. Therefore, if we substitute this value of A in step 7, we will have the following equation: K2 = (gX)Y mod n = gXY' mod n. Now, basic mathematics says that: KYX = KXY Therefore, in this case, we have: K I = K2 = K.

1. Firstly, Alice and Bob agree on two large prime numbers, n and g. These two integers need not be kept secret. Alice and Bob can use an insecure channel to agree on them.
2. Alice chooses another large random number x, and calculates A such that:
   $A = g^x \bmod n$
3. Alice sends the number A to Bob.
4. Bob independently chooses another large random integer y and calculates B such that:
   $B = g^y \bmod n$
5. Bob sends the number B to Alice.
6. A now computes the secret key K1 as follows:
   $K1 = B^x \bmod n$
7. B now computes the secret key K2 as follows:
   $K2 = A^y \bmod n$

Fig 3.2: Diffie- Hellman Key Exchange algorithm

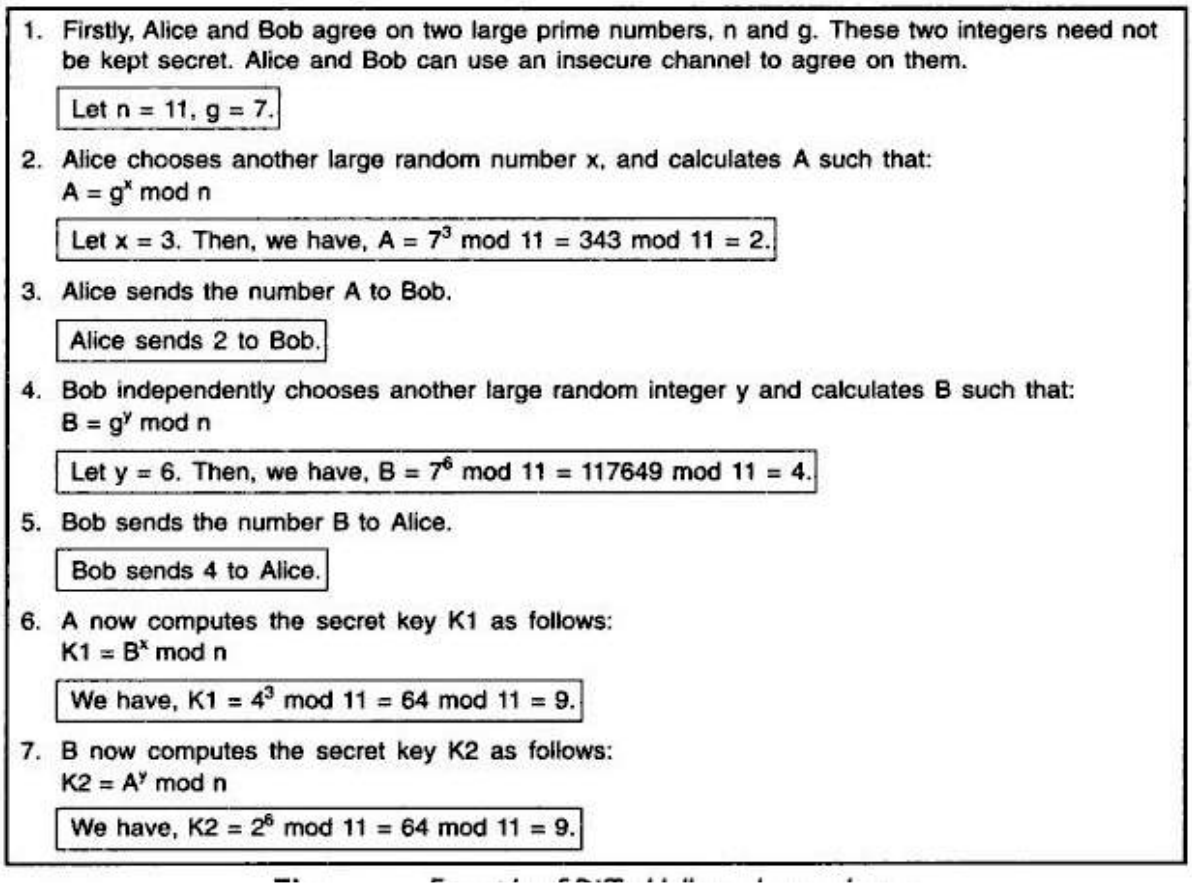


Fig 3.3: Example of Diffie-Hellman Key exchange

3.3 KEY MANAGEMENT:

Key management refers to management of cryptographic keys in a cryptosystem. This includes dealing with the generation, exchange, storage, use and replacement of keys. Once keys are inventoried, key management typically consists of three steps: exchange, storage and use. Key exchange (also key establishment) is any method in cryptography by which cryptographic keys are exchanged between two parties, allowing use of a cryptographic algorithm. If the cipher is a symmetric key cipher, both will need a copy of the same key. If an asymmetric key cipher with the public/private key property, both will need the other's public key.

3.3.1: Key storage: However distributed, keys must be stored securely to maintain communications security. Security is a big concern[4] and hence there are various techniques in use to do so. Likely the most common is that an encryption application manages keys for the user and depends on an access password to control use of the key.

Key use : The major issue is length of time a key is to be used, and therefore frequency of replacement. Because it increases any attacker's required effort, keys should be frequently changed. This also limits loss of information, as the number of stored encrypted messages which will become readable when a key is found will decrease as the frequency of

key change increases.

Challenges

Several challenges IT organizations face when trying to control and manage their encryption keys are:

1. Scalability: Managing a large number of encryption keys.

2. Security: Vulnerability of keys from outside hackers, malicious insiders.

3. Availability: Ensuring data accessibility for authorized users.

4. Heterogeneity: Supporting multiple databases, applications and standards.

5. Governance: Defining policy-driven access control and protection for data

Distribution of public Keys:

Several techniques have been proposed for the distribution of public keys.

Virtually all these proposals can be grouped into the following general schemes:

• Public announcement • publicly available directory

• Public-key authority. • Public-key certificates

3.3.2: PUBLIC ANNOUNCEMENT OF PUBLIC KEYS:

On the face of it, the point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large (Refer fig below)

Figure Uncontrolled Public-Key Distribution

Fig 3.4: Uncontrolled public key distributions

Publicly Available Directory

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

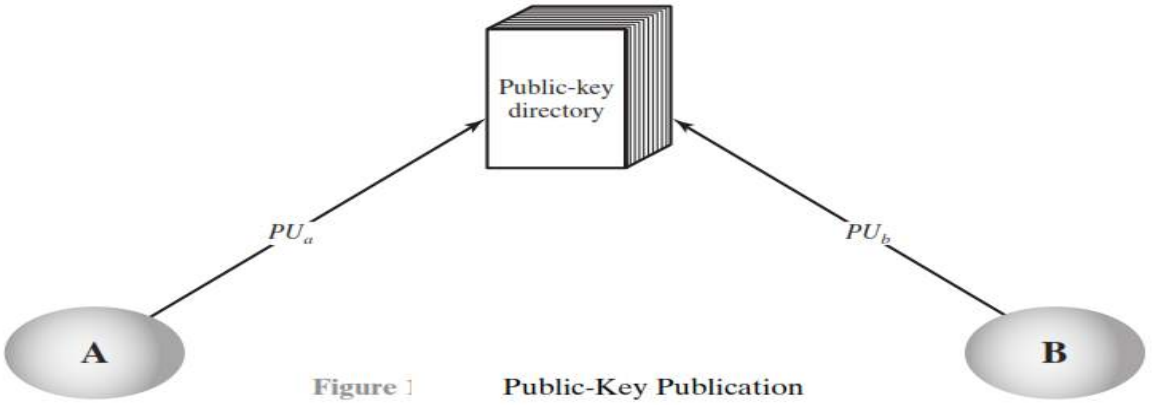


Figure 1 Public-Key Publication

This scheme is clearly more secure than individual public announcements

Fig 3.5: Public-Key Publication

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

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

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

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

3.3.3: **PUBLIC-KEY AUTHORITY:**

Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. A typical scenario is illustrated in Figure below.

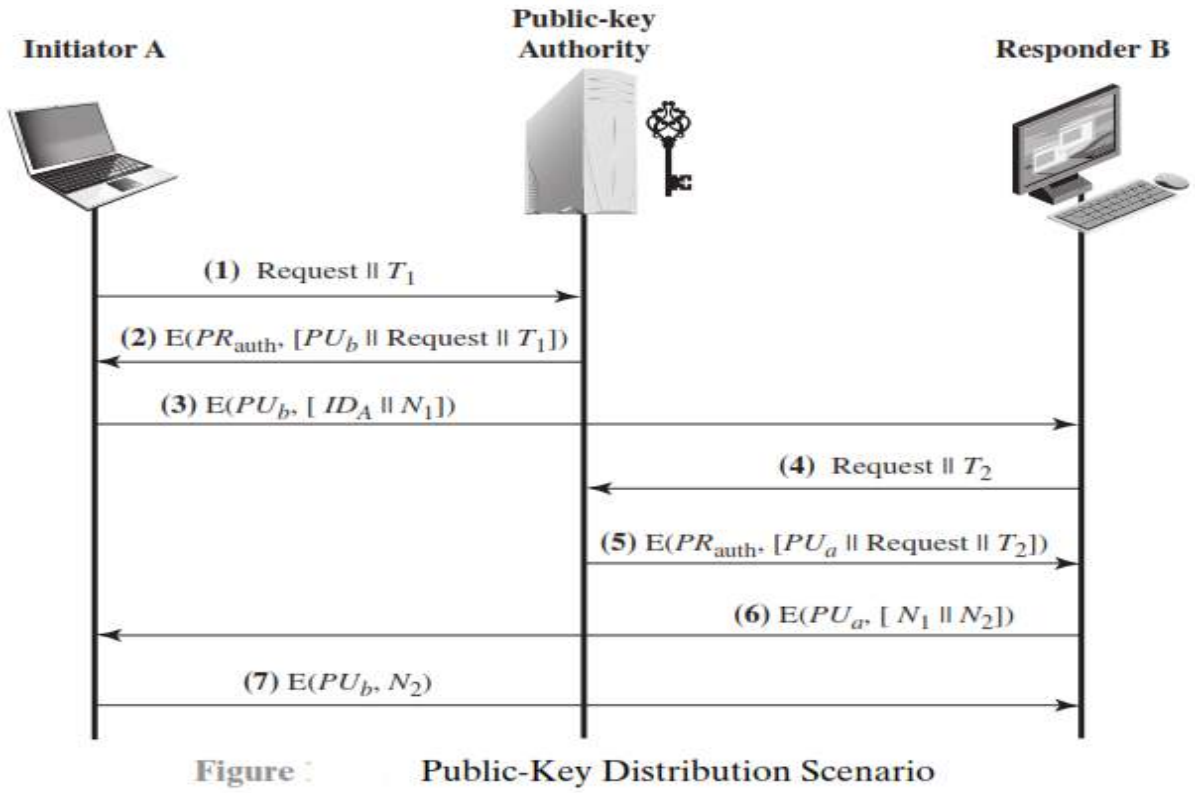


Figure : Public-Key Distribution Scenario

Fig 3.6: Public Key Distribution Scenario

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

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

• B's public key, PUb, which A can use to encrypt messages destined for B

• The original request used to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority

• The original timestamp given so A can determine that this is not an old message from the authority containing a key other than B's current public key

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

4, 5. B retrieves A's public key from the authority in the same manner as A retrieved B's public key. At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:

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

7. A returns N2, which is encrypted using B's public key, to assure B that its correspondent is A.

Thus, a total of seven messages are required.

3.4: PUBLIC-KEY CERTIFICATES:

The public key certificate relies on certificates that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party.

Typically, the third party is a certificate authority, such as a government agency or a financial institution that is trusted by the user community. A user can present his or her public key to the authority in a secure manner and obtain a certificate. The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature.

We can place the following requirements on this scheme:

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.

2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.

3. Only the certificate authority can create and update certificates.

4. Any participant can verify the currency of the certificate.

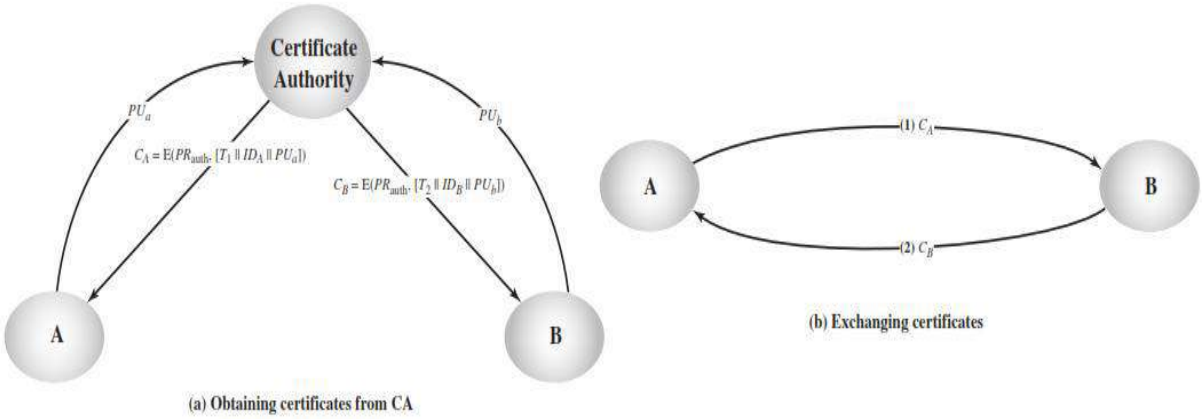


Figure Exchange of Public-Key Certificates

Fig 3.7: Exchange of public key exchange

A certificate scheme is illustrated in Figure above. Each participant applies to the certificate authority, supplying a public key and requesting a certificate. Application must be in person or by some form of secure authenticated communication.
For participant A, the authority provides a certificate of the form

$$C_A = \mathrm{E}(PR_{\mathrm{auth}}, [T \| ID_A \| PU_a])$$

where PRauth is the private key used by the authority and T is a timestamp. A may

then pass this certificate on to any other participant, who reads and verifies the certificate

as follows:

$$D(PU_{auth}, C_A) = D(PU_{auth}, E(PR_{auth}, [T \| ID_A \| PU_a])) = (T \| ID_A \| PU_a)$$

The recipient uses the authority's public key, PUauth, to decrypt the certificate. Because the certificate is readable only using the authority's public key, this verifies that the certificate came from the certificate authority. The elements IDA and PUa provide the recipient with the name and public key of the certificate's holder. The timestamp T validates the currency of the certificate.

One scheme has become universally accepted for formatting public-key certificates: the X.509 standard. X.509 certificates are used in most network security applications, including IP security, transport layer security (TLS), and S/MIME.

3.4.1: Message digest:

Message digest (also called as hash) is a fingerprint or the summary of a message. It is used to verify the integrity of the data (i.e. to ensure that a message has not been tampered with after it leaves the sender but before it reaches the receiver).

Idea of a Message Digest:

The concept of message digests is based on similar principles. However, it is slightly wider in scope. For instance, suppose we have a number 4000 and we divide it by 4 to get 1000, 4 becomes a fingerprint of the number 4000. Dividing 4000 by 4 will always yield 1000. If we change either 4000 or 4, the result will not be 1000. Another important point is, if we are simply given the number 4, but are not given any further information, we would not be able to trace back the equation 4 x 1000 =4000. Thus, we have one more important concept here. The fingerprint of a message (in this case, the number 4) does not tell anything about the original message (in this case, the number 4000). This is because there are infinite other possible equations, which can produce the result 4.

Another simple example of message digest is shown in Fig. below.

- **Original number is 7391743**

Operation	Result
Multiply 7 by 3	21
Discard first digit	1
Multiply 1 by 9	9
Multiply 9 by 1	9
Multiply 9 by 7	63
Discard first digit	3
Multiply 3 by 4	12
Discard first digit	2
Multiply 2 by 3	6

- **Message digest is 6**

Fig 3.8: Example of message digest

Let us assume that we want to calculate the message digest of a number

7391753. Then, we multiply each digit in the number with the next digit (excluding it if it is

0) and disregarding the first digit of the multiplication operation, if the result is a two-digit

number.

3.4.2: Requirements of a Hash function (Message digest):

1. It should be a one way function. That means given the message it should be easy to find out its digest, and the reverse should be impossible.(getting back the message from its digest must be infeasible)

2. No two different messages should produce a same digest. This requirement is stated as collision free property.

3.4.3: MD5 :

MD5 is a message digest algorithm developed by Ron Rivest. MD5 is quite fast and produces 128-bit message digests. Over the years, researchers have developed potential weaknesses in MD5. However, so far, MD5 has been able to successfully defend itself against collisions. This may not be guaranteed for too long, though.

After some initial processing, the input text is processed in 512-bit blocks (which are further divided into 16 32-bit sub-blocks). The output of the algorithm is a set of four 32-bit blocks, which make up the 128-bit message digest.

3.5.4 How MD5 Works?

Step 1: Padding :The first step in MD5 is to add padding bits to the original message. The aim of this step is to make the length of the original message equal to a value, which is 64 bits less than an exact multiple of 512. For example, if the length of the original message is 1000 bits, we add a padding of 472 bits to make the length of the message 1472 bits. This is because, if we add 64 to 1472, we get 1536, which is a multiple of 512 (because 1536 = 512 x 3).

Thus, after padding, the original message will have a length of 448 bits (64 bits less than 512), 960 bits (64 bits less than 1024), 1472 bits (64 bits less than 1536), etc. The padding consists of a single 1-bit, followed by as many 0-bits, as required. Note that padding is always added, even if the message length is already 64 bits less than a multiple of 512. Thus, if the message were already of length say 448 bits, we will add a padding of 512 bits to make its length 960 bits. Thus, the padding length is any value between 1 and 512.

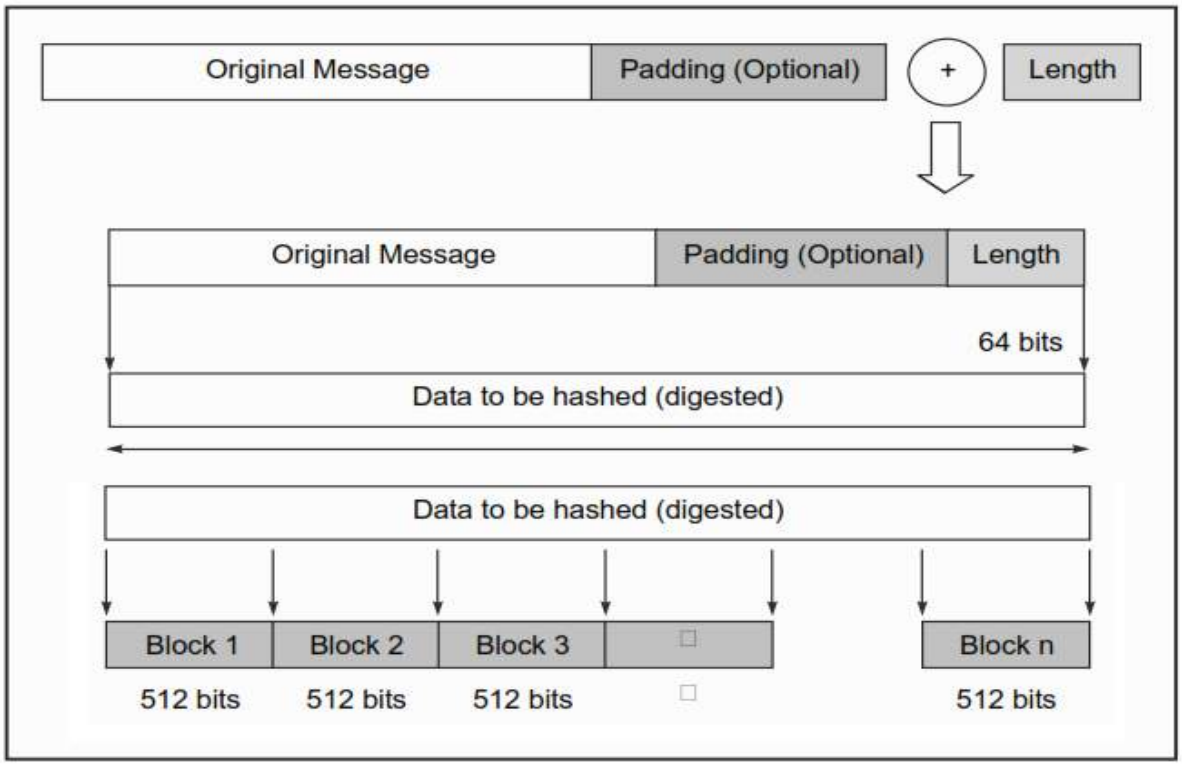


Fig 3.9: Message Digest

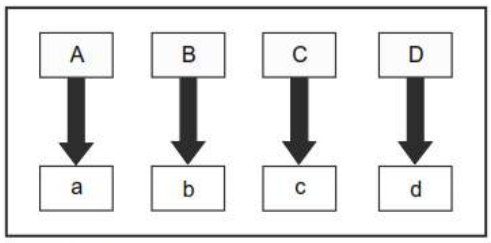
Step 2: Append length After padding bits are added, the next step is to calculate the original length of the message and add it to the end of the message, after padding. The length of the message is calculated, excluding the padding bits. This length of the original message is now expressed as a 64-bit value and these 64 bits are appended to the end of the original message + padding.

Step 3: Divide the input into 512-bit blocks Now, we divide the input message into blocks, each of length 512 bits. (Refer fig)

Step 4: Initialize chaining variables In this step, four variables (called as chaining variables) are initialized. They are called as A, B, C and D. Each of these is a 32-bit number. The initial hexadecimal values of these chaining variables are shown in Fig. below.

A	Hex	01	23	45	67
B	Hex	89	AB	CD	EF
C	Hex	FE	DC	BA	98
D	Hex	76	54	32	10

Fig. *Chaining variables*



Copying chaining variables into temporary variables

Fig 3.10: Chaining Variables

ep 5: Process blocks:

Copy the four chaining variables into four corresponding variables, a, b, c and d. After all the initializations, the real algorithm begins. There is a loop that runs for as many 512-bit blocks as are in the message. Now, we have four rounds. In each round, we process all the 16 sub-blocks belonging to a block. The inputs to each round are: (a) all the 16 sub-blocks, (b) the variables a, b, c, d and (c) some constants, designated as t.

All the four rounds vary in one major way: Step 1 of the four rounds has different processing. The other steps in all the four rounds are the same. • In each round, we have 16 input sub-blocks, named M[0], M[1], …, M[15] or in general, M[i], where i varies from 0 to 15. As we know, each sub-block consists of 32 bits.

• Also, t is an array of constants. It contains 64 elements, with each element consisting of 32 bits. We denote the elements of this array t as t[1], t[2], … t[64] or in general as t[k], where k varies from 1 to 64. Since there are four rounds, we use 16 out of the 64 values of t in each round.

Let us summarize these iterations of all the four rounds. In each case, the output of the intermediate as well as the final iteration is copied into the register abcd. Note that we have 16 such iterations in each round.

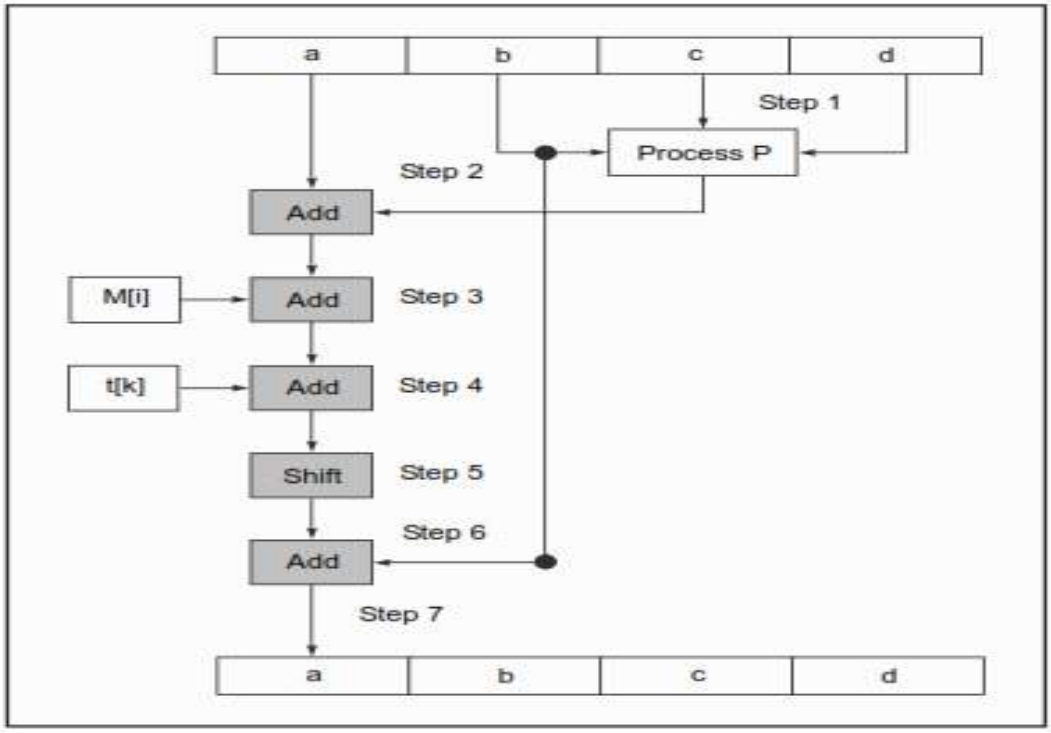


Fig. *One MD5 operation*

Fig 3.11: One MD5 operation

1. A process P is first performed on b, c and d. This process P is different in all the four rounds.
2. The variable a is added to the output of the process P (i.e. to the register abcd).
3. The message sub-block M[i] is added to the output of Step 2 (i.e. to the register abcd).
4. The constant t[k] is added to the output of Step 3 (i.e. to the register abcd).
5. The output of Step 4 (i.e. the contents of register abcd) is circular-left shifted by s bits. (The value of s keeps changing).
6. The variable b is added to the output of Step 5 (i.e. to the register abcd).
7. The output of Step 6 becomes the new abcd for the next step.

3. 5 : SECURE HASH ALGORITHM (SHA):

The National Institute of Standards and Technology (NIST) along with NSA developed the Secure Hash Algorithm (SHA). In 1993. SHA is a modified version of

MD5 and its design closely resembles MD5. SHA works with any input message that is less than 2

64 bits in length. The output of SHA is a message digest, which is 160 bits in length (32 bits more than the message digest produced by MD5). The word Secure in SHA was decided based on two features. SHA is designed to be computationally infeasible to:

(a) Obtain the original message, given its message digest and

(b) Find two messages producing the same message digest

Step1 to 3 MD5 and SHA are same. In step 4 and process SHA differs from MD5.

Step 4: Initialize chaining variables Now, five chaining variables A through E are initialized. Remember that we had four chaining variables, each of 32 bits in MD5 (which made the total length of the variables 4 x 32 = 128 bits). Recall that we stored the intermediate as well as the final results into the combined register made up of these four

chaining variables, i.e. abcd. Since in the case of SHA, we want to produce a message digest of length 160 bits, we need to have five chaining variables here (5x32 = 160 bits). In SHA, the variables A through D have the same values as they had in MD5. Additionally, E is initialized to Hex C3 D2 E1 F0.

Step 5: Process blocks Now the actual algorithm begins. Here also, the steps are quitesimilar to those in MD5. SHA has four rounds, each round consisting of 20 steps. Each round takes the current 512- bit block, the register abcde and a constant K[t] (where t = 0 to 79) as the three inputs. It then updates the contents of the register abcde using the SHA algorithm steps. Also notable is the fact that we had 64 constants defined as t in MD5. Here, we have only four constants defined for K[t], one used in each of the four rounds.

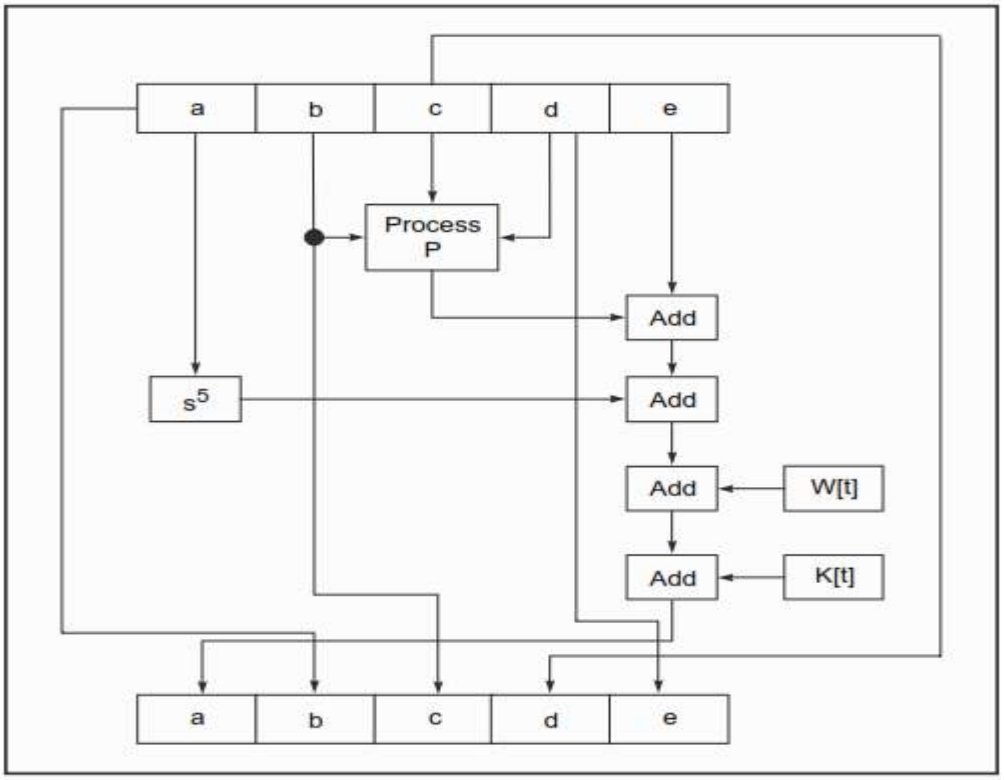


Fig. *Single SHA-1 iteration*

Fig. 3.12: Single SHA-1 iteration

Table 3.1: Comparison of MD5 and SHA-1

Table *Comparison of MD5 and SHA-1*

Point of discussion	*MD5*	*SHA-1*
Message digest length in bits	128	160
Attack to try and find the original message given a message digest	Requires 2^{128} operations to break in	Requires 2^{160} operations to break in, therefore more secure
Attack to try and find two messages producing the same message digest	Requires 2^{64} operations to break in	Requires 2^{80} operations to break in
Successful attacks so far	There have been reported attempts to some extent (as we discussed earlier)	No such claims so far
Speed	Faster (64 iterations and 128-bit buffer)	Slower (80 iterations and 160-bit buffer)
Software implementation	Simple, does not need any large programs or complex tables	Simple, does not need any large programs or complex tables

3.6 MESSAGE AUTHENTICATION CODE (MAC):

The concept of Message Authentication Code (MAC) is quite similar to that of a message digest. However, there is one difference. As we have seen, a message digest is simply a fingerprint of a message. There is no cryptographic process involved in the case of message digests. In contrast, a MAC requires that the sender and the receiver should know a shared symmetric (secret) key, which is used in the preparation of the MAC. Thus, MAC involves cryptographic processing. Let us assume that the sender A wants to send a message M to a receiver B. How the MAC processing works is shown in Fig. below.

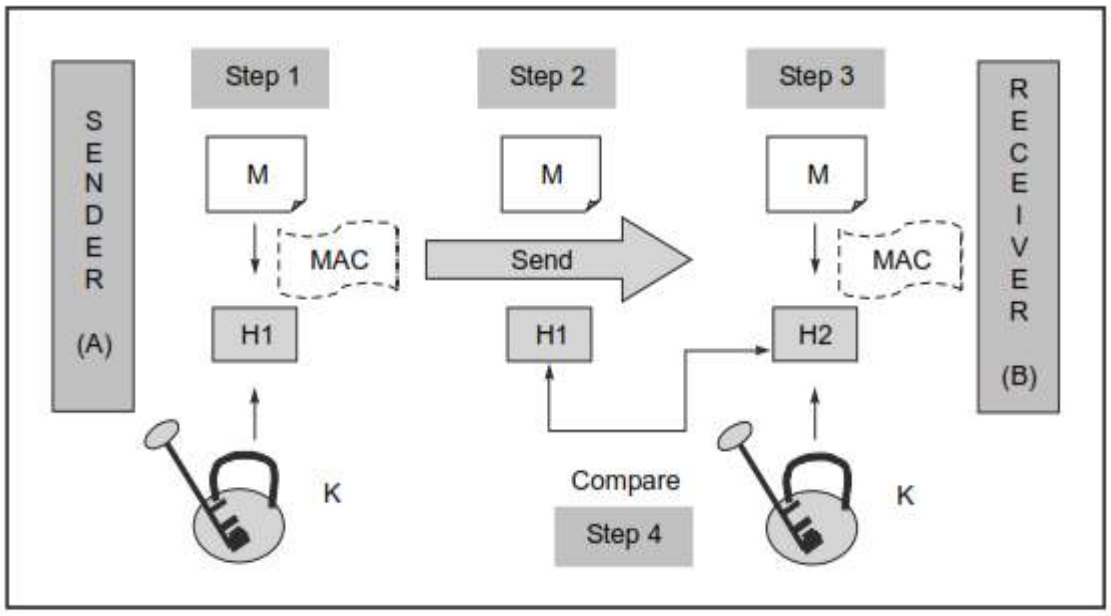


Fig.3.13: Message Authentication Code

1. A and B share a symmetric (secret) key K, which is not known to anyone else. A calculates the MAC by applying key K to the message M.

2. A then sends the original message M and the MAC H1 to B.

3. When B receives the message, B also uses K to calculate its own MAC H2 over M.

4. B now compares H1 with H2. If the two match, B concludes that the message M has not been changed during transit. However, if H1 ≠ H2, B rejects the message, realizing that the message was changed during transit.

3.7 HMAC

HMAC stands for Hash-based Message Authentication Code. HMAC has been chosen as a mandatory security implementation for the Internet Protocol (IP) security and is also used in the Secure Socket Layer (SSL) protocol, widely used on the Internet. The fundamental idea behind HMAC is to reuse the existing message digest algorithms, such as MD5 or SHA-1.It treats the message digest as a black box. Additionally, it uses the shared symmetric key to encrypt the message digest, which produces the output MAC. This is shown in figure below.

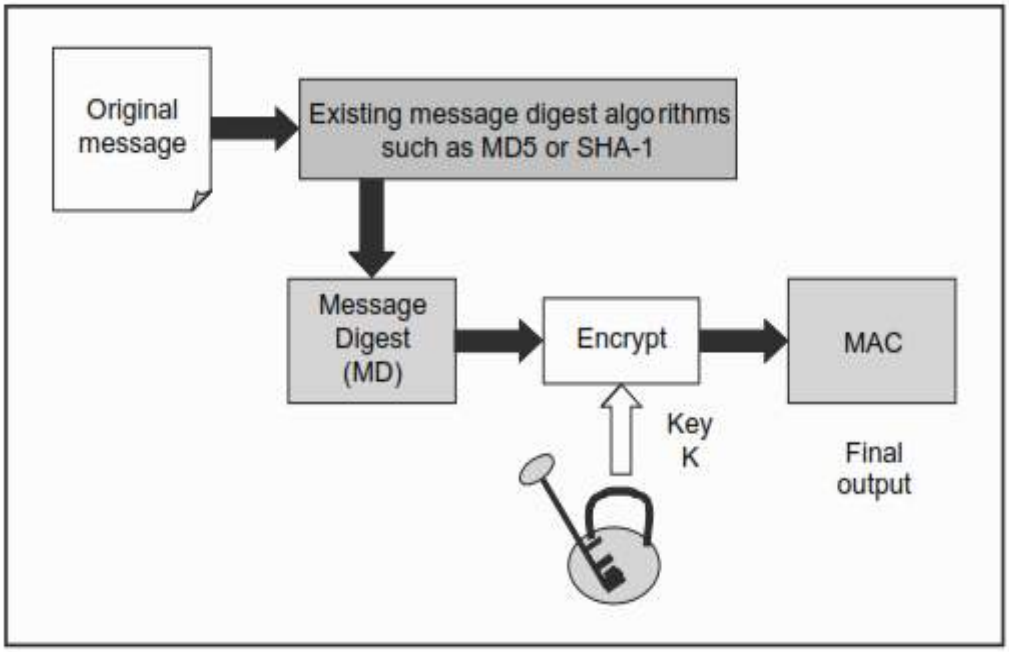


Fig.3.14: HMAC Concept

How HMAC Works?

Let us now take a look at the internal working of HMAC. For this, let us start with the various variables that will be used in our HMAC discussion. MD = The message digest/hash function used (e.g. MD5, SHA-1, etc.) M = The input message whose MAC is to be calculated L = The number of blocks in the message M b = The number of bits in each block K = The shared symmetric key to be used in HMAC ipad = A string 00110110 repeated b/8 times opad = A string 01011010 repeated b/8 times

Step 1: Make the length of K equal to b

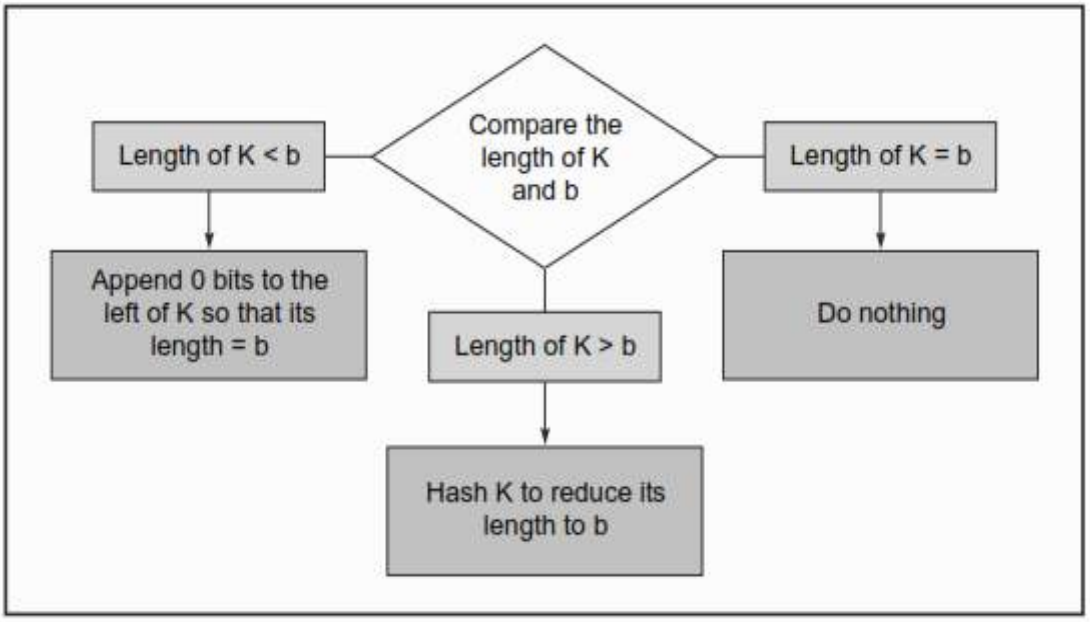


Fig. 3.14: Step1 of HMAC

Step 2: XOR K with ipad to produce S1 We XOR K (the output of Step 1) and ipad to produce a variable called as S1.

Step 3: Append M to S1 We now take the original message (M) and simply append it to the end of S1 (which was calculated in Step 2).

Step 4: Message digest algorithm Now, the selected message digest algorithm (e.g. MD5, SHA-1, etc) is applied to the output of Step 3 (i.e. to the combination of S1 and M). Let us call the output of this operation as H.

Step 5: XOR K with opad to produce S2 Now, we XOR K (the output of Step 1) with opad to produce a variable called as S2.

Step 6: Append H to S2 In this step, we take the message digest calculated in step 4 (i.e. H) and simply append it to the end of S2 (which was calculated in Step 5).

Step 7: Message digest algorithm Now, the selected message digest algorithm (e.g. MD5, SHA-1, etc) is applied to the output of Step 6 (i.e. to the concatenation of S2 and H). This is the final MAC that we want.

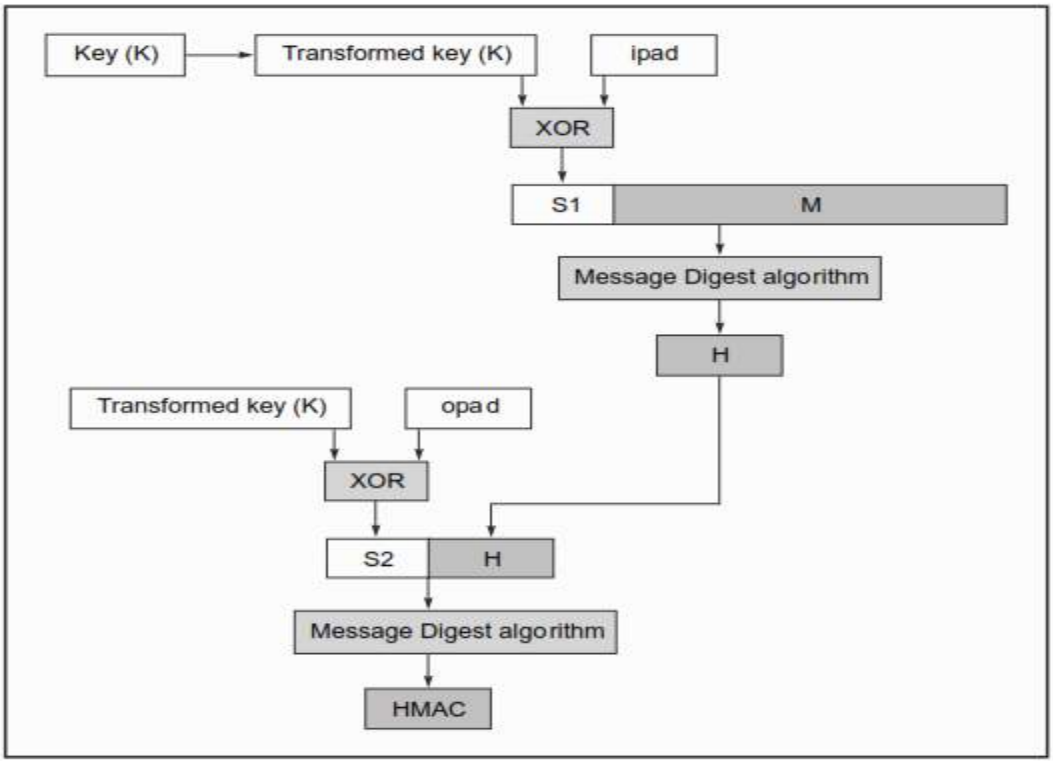


Fig.3.15: Complete HMAC Operation

3.8: DIGITAL SIGNATURES:

These are the public-key primitives of message authentication. In the physical world, it is common to use handwritten signatures on handwritten or typed messages. They are used to bind signatory to the message. Similarly, a digital signature is a technique that binds a person/entity to the digital data. This binding can be independently verified by receiver as well as any third party. Digital signature is a cryptographic value that is calculated from the data and a secret key known only by the signer.

The exchange of data is authenticated by signing a mutually obtainable hash; each party encrypts the hash with its private key. The hash is generated over important parameters, such as user IDs and nonces. In real world, the receiver of message needs assurance that the message belongs to the sender and he should not be able to repudiate the origination of that message. This requirement is very crucial in business applications, since likelihood of a dispute over exchanged data is very high.

As mentioned earlier, the digital signature scheme is based on public key cryptography. The model of digital signature scheme is depicted in the following illustration.

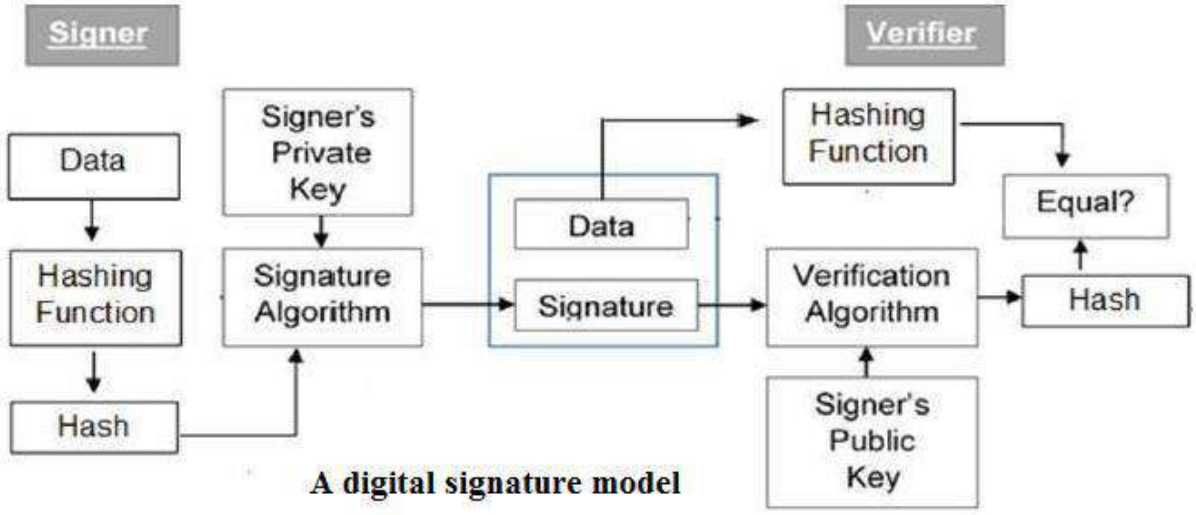


Fig 3.16: A digital Signature Model

The following points explain the entire process in detail −

- Each person adopting this scheme has a public-private key pair.
- Generally, the key pairs used for encryption/decryption and signing/verifying are different. The private key used for signing is referred to as the signature key and the public key as the verification key.
- Signer feeds data to the hash function and generates hash of data.
- Hash value and signature key are then fed to the signature algorithm which produces the digital signature on given hash. Signature is appended to the data and then both are sent to the verifier. Verifier feeds the digital signature and the verification key into the verification algorithm. The verification algorithm gives some value as output.
- Verifier also runs same hash function on received data to generate hash value.
- For verification, this hash value and output of verification algorithm are compared. Based on the comparison result, verifier decides whether the digital signature is valid.
- Since digital signature is created by 'private' key of signer and no one else can have this key; the signer cannot repudiate signing the data in future.

It should be noticed that instead of signing data directly by signing algorithm, usually a hash of data is created. Since the hash of data is a unique representation of data, it is sufficient to sign the hash in place of data. The most important reason of using hash instead of data directly for signing is efficiency of the scheme.

Let us assume RSA is used as the signing algorithm. As discussed in public key encryption chapter, the encryption/signing process using RSA involves modular exponentiation.

Signing large data through modular exponentiation is computationally expensive and time consuming. The hash of the data is a relatively small digest of the data, hence signing a hash is more efficient than signing the entire data.

3.8.1: DIGITAL SIGNATURE REQUIREMENTS

We can formulate the following requirements for a digital signature.

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

3.9: KERBEROS

Many real-life systems use an authentication protocol called as Kerberos, to allow the workstations to allow network resources in a secure manner. The name Kerberos signifies a multi-headed dog in the Greek mythology (apparently used to keep outsiders away). Version 4 of Kerberos is found in most practical. implementations. However, Version 5 is also in use now.

3.9.1 How does Kerberos Work?

There are four parties involved in the Kerberos protocol:

Alice: The client workstation

Authentication Server (AS): Verifies (authenticates) the user during login

Ticket Granting Server (TGS): Issues tickets to certify proof of identity

Bob: The server offering services such as network printing, file sharing or an application program

The job of AS is to authenticate every use at the login time. AS shares a unique secret password with every user.

The job of TGS is to certify to the servers in the network that a user is really what she claims to be. For proving this, the mechanism of tickets (which allow entry into a

Server, just as a ticket allows parking a car or entering a music concert) is used.

Step 1: Login To start with, Alice, the user, sits down at an arbitrary public workstation and enters her name. The work station sends her name in plain text to the AS.

In response, the AS performs several actions. It first creates a package of the user name (Alice) and a randomly generated session key (KS). It encrypts this package with the symmetric key that the AS shares with the Ticket Granting Server (TGS). The output of

this step is called as the Ticket Granting Ticket (TGT). Note that the TGT can be opened

only by the TGS, since only it possesses the corresponding symmetric key for decryption.

The AS then combines the TGT with the session key (KS), and encrypts the two together using a symmetric key derived from the password of Alice (KA). Note that the final output can, therefore, be opened only by Alice.

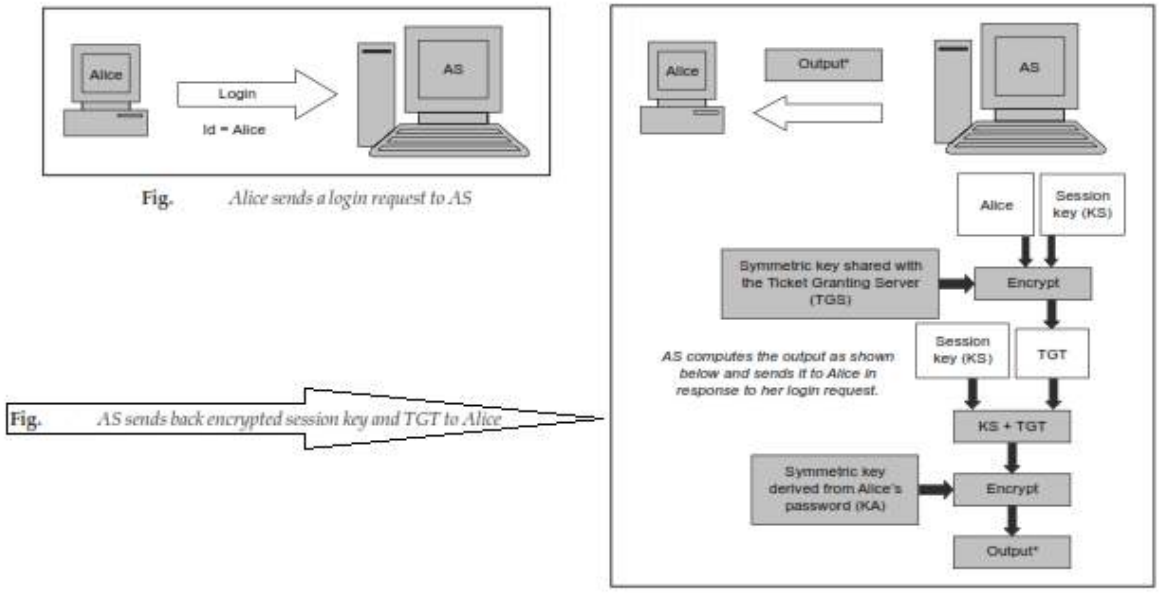


Fig 3.17: Kerberos

Step 2: Obtaining a service granting ticket (SGT) Now, let us assume that after a successful login, Alice wants to make use of Bob – the email server, for some email communication. For this, Alice would inform her workstation that she needs to contact Bob. Therefore, Alice needs a ticket to communicate with Bob. At this juncture, Alice's workstation creates a message intended for the Ticket Granting Server (TGS), which contains the following items:

• The TGT as in step 1

• The id of the server (Bob) whose services Alice is interested in

• The current timestamp, encrypted with the same session key (KS)

As we know, the TGT is encrypted with the secret key of the Ticket Granting Server (TGS). Therefore, only the TGS can open it. This also serves as a proof to the TGS that the message indeed came from Alice. Why? This is because, if you remember, the TGT was created by the AS (remember that only the AS and the TGS know the secret key of TGS). Furthermore, the TGT and the KS were encrypted together by the AS with the secret key derived from the password of Alice. Therefore, only Alice could have opened that package and retrieved the TGT. Once the TGS is satisfied of the credentials of Alice, the TGS creates a session key KAB, for Alice to have secure communication with Bob. TGS sends it twice to Alice: once combined with Bob's id (Bob) and encrypted with the session key (KS) and a second time, combined with Alice's id (Alice) and encrypted with Bob's secret key (KB). This is shown in Fig. below.

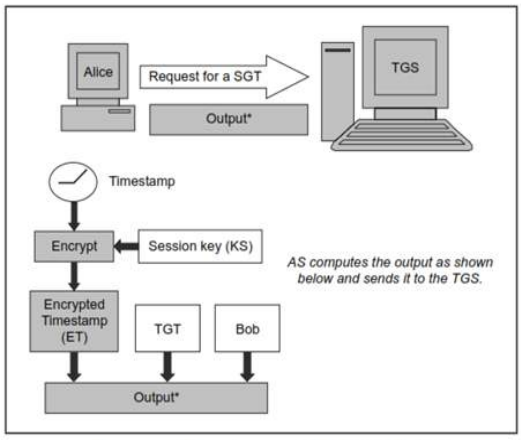


Fig. *Alice sends a request for a SGT to the TGS*

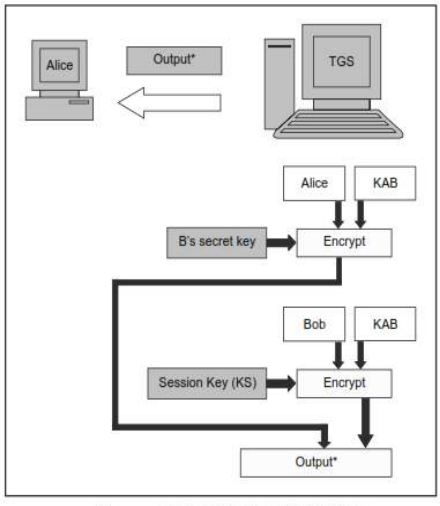


Fig. *TGS sends response back to Alice*

Fig 3.8: Kerberos Working

Step 3: User contacts Bob for accessing the server Alice can now send KAB to Bob in order to enter into a session with him. Since this exchange is also desired to be secure, Alice can simply forward KAB encrypted with Bob's secret key (which she had received from the TGS in the previous step) to Bob. This will ensure that only Bob can access KAB. Furthermore, to guard against replay attacks, Alice also sends the timestamp, encrypted with KAB to Bob. Since only Bob has his secret key, he uses it to first obtain the information (Alice + KAB). From this, it gets the key KAB, which he uses to decrypt the encrypted timestamp value.

Now how would Alice know if Bob received KAB correctly or not? In order to satisfy this query, Bob now adds 1 to the timestamp sent by Alice, encrypts the result with KAB and sends it back to Alice. This is shown in Fig. above. Since only Alice and Bob know KAB, Alice can open this packet and verify that the timestamp incremented by Bob was indeed the one sent by her to Bob in the first place. Now, Alice and Bob can communicate securely with each other. They would use the shared secret key KAB to encrypt messages before sending and also to decrypt the encrypted messages received from each other.

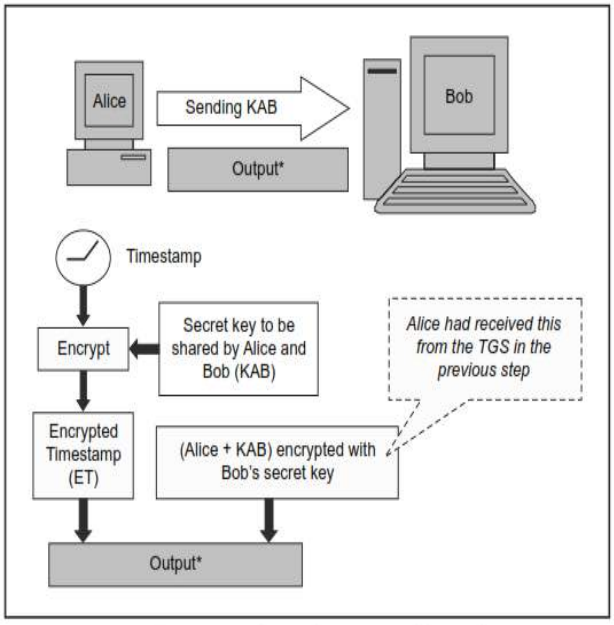


Fig. *Alice sends KAB securely to Bob*

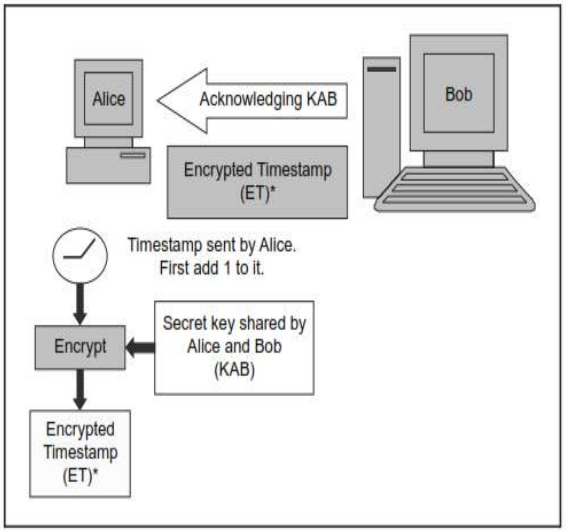


Fig. *Bob acknowledges the receipt of KAB*

Fig.3.19: Acknowledgement

3.10: X.509 CERTIFICATES

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 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

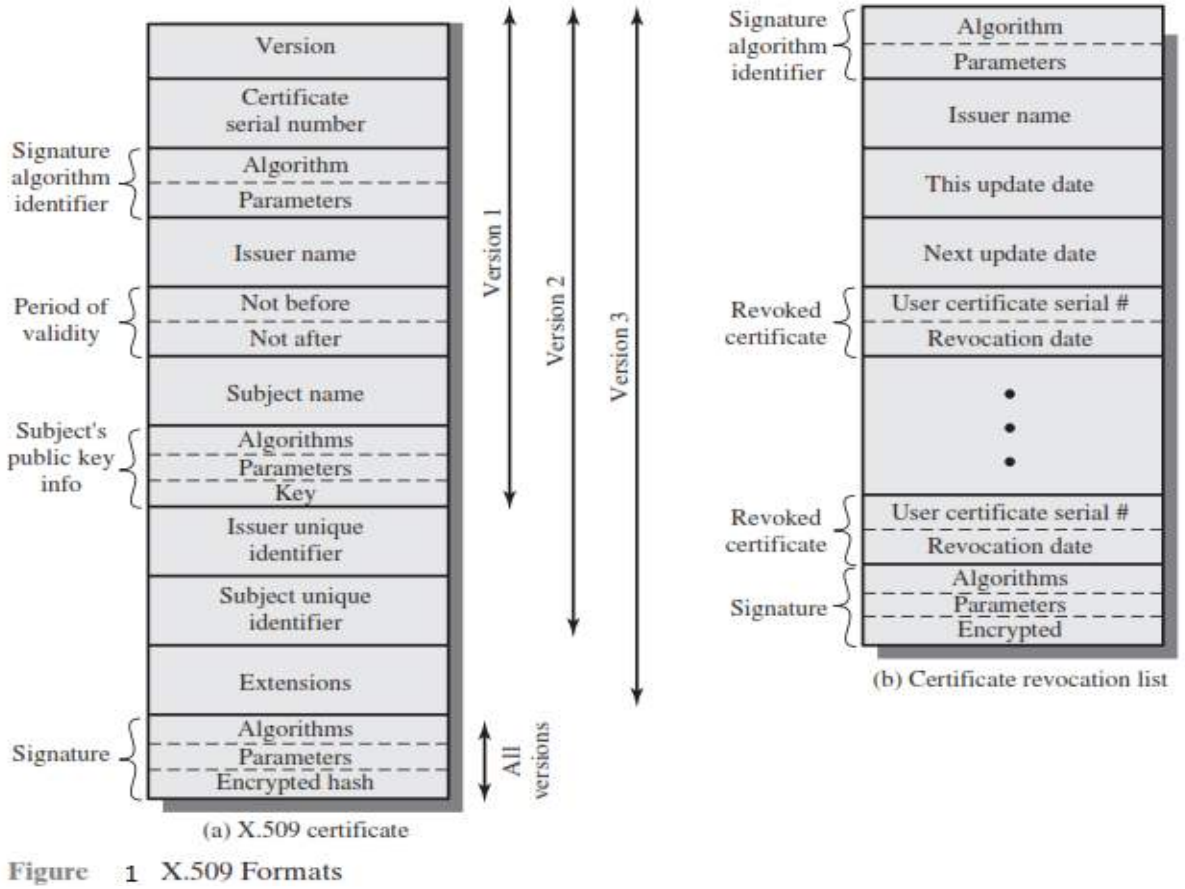


Figure 1 X.509 Formats

Fig 3.20: General form of a certificate

The above figure shows the general format of a certificate, which includes the following elements.

• **Version:** Differentiates among successive versions of the certificate format; the default is version 1. If the Issuer Unique Identifier or Subject Unique Identifier are present, the value must be version 2. If one or more extensions are present, the version must be version 3.

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

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

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

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

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

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

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

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

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

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

3.11: PUBLIC-KEY INFRASTRUCTURE:

Public-key infrastructure (PKI) is the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography. The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys. The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the driving force behind setting up a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet. This section describes the PKIX model.

• **End entity:** A generic term used to denote end users, devices (e.g., servers, routers), or any other entity that can be identified in the subject field of a public key certificate. End entities typically consume and/or support PKI-related services.

• **Certification authority (CA):** The issuer of certificates and (usually) certificate revocation lists (CRLs). It may also support a variety of administrative functions, although these are often delegated to one or more registration authorities.

• **Registration authority (RA):** An optional component that can assume a number of administrative functions from the CA. The RA is often associated with the end entity registration process, but can assist in a number of other areas as well

. • **CRL issuer:** An optional component that a CA can delegate to publish CRLs.

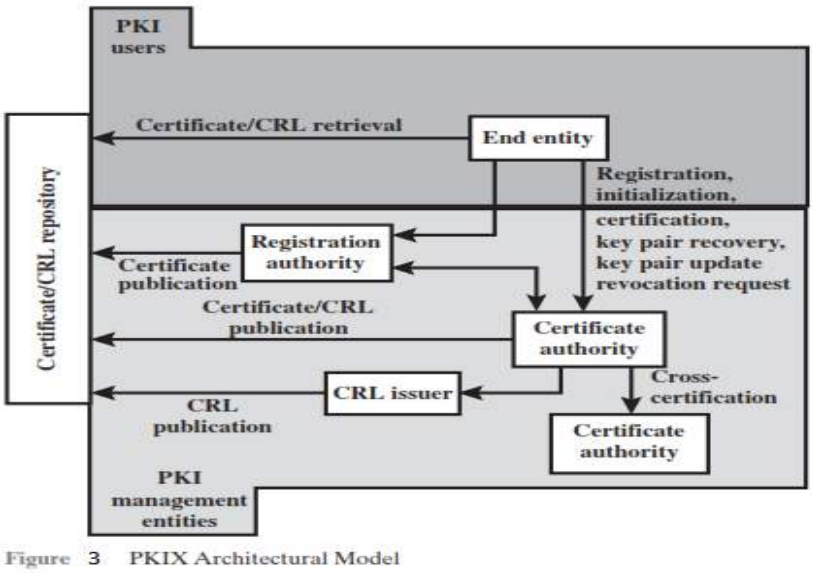


Figure 3 PKIX Architectural Model

Fig. 3.21: PKIX Architectural Model

Figure 3.21 shows the interrelationship among the key elements of the PKIX model. These elements are

• **Repository:** A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by end entities. PKIX Management Functions PKIX identifies a number of management functions that potentially need to be supported by management protocols. These are indicated in Figure 3 and include the following:

• **Registration:** This is the process whereby a user first makes itself known to a CA (directly, or through an RA), prior to that CA issuing a certificate or certificates for that user. Registration begins the process of enrolling in a PKI. Registration usually involves some off-line or online procedure for mutual authentication. Typically, the end entity is issued one or more shared secret keys used for subsequent authentication.

• **Initialization:** Before a client system can operate securely, it is necessary to install key materials that have the appropriate relationship with keys stored elsewhere in the infrastructure. For example, the client needs to be securely initialized with the public key and other assured information of the trusted CA(s) to be used in validating certificate paths.

• **Certification:** This is the process in which a CA issues a certificate for a user's public key and returns that certificate to the user's client system and/or posts that certificate in a repository.

• **Key pair recovery:** Key pairs can be used to support digital signature creation and verification, encryption and decryption, or both. When a key pair is used for encryption/decryption, it is important to provide a mechanism to recover the necessary decryption keys when normal access to the keying material is no longer possible, otherwise it

will not be possible to recover the encrypted data. Loss of access to the decryption key can result from forgotten passwords/PINs, corrupted disk drives, damage to hardware tokens, and so on. Key pair recovery allows end entities to restore their encryption/decryption key pair from an authorized key backup facility (typically, the CA that issued the end entity's certificate).

• **Key pair update:** All key pairs need to be updated regularly (i.e., replaced with a new key

pair) and new certificates issued. Update is required when the certificate lifetime expires and

as a result of certificate revocation.

• **Revocation request:** An authorized person advises a CA of an abnormal situation requiring certificate revocation. Reasons for revocation include private key compromise, change in affiliation, and name change.

• **Cross certification:** Two CAs exchange information used in establishing a cross certificate.

A cross-certificate is a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.

Review Questions:

Part-A

1. List Out the applications of public Key Cryptosystem
2. Define Hash Function
3. Differentiate MD5 and SHA1
4. Analyze MAC Function
5. Define Digital signature and list out its uses
6. Differentiate firewall and Kerberos
7. List out the advantages and disadvantages of Kerberos
8. Discuss public key cryptography
9. Sketch the basic entities of Public key Infrastructure
10. List out the Key Management steps
11. What are the properties a digital signature scheme satisfy?
12. List the basic arithmetic and logical functions used in MD5

Part-B

1. Explain Difffie Helman key exchange algorithm in detail
2. Create a Message Authentication code and explain the procedure for checking the integrity of the message
3. Explain the Various Hash algorithms in detail
4. Analyze the generation of digital signature and its various applications
5. Explain Kerberous and X.509 authentication service

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT - 4

SCS1316 - NETWORK SECURITY

UNIT- IV INTERNET SECURITY

Syllabus:

Email Security-PGP-S/MIME- Secured Electronic Transaction-IP Security Overview- IPSec Documents-IPSec Services, IPSec Architecture, IP Traffic Processing-Encapsulating Security Payload-Internet key Exchange- Firewalls- Stateful Packet Inspection- Application Gateways/Proxies- Hybrid Systems

4.1 Electronic Mail Security

Overview

- Pretty Good Privacy (PGP)
- S/MIME
- DomainKeys Identified Mail (DKIM)

4.2 Email Security Enhancements

- Confidentiality: Protection from disclosure
- Authentication: Of sender of message
- Message integrity: Protection from modification
- Non-repudiation of origin: Protection from denial by sende**r**

4.1 PGP – Authentication and Confidentiality

2013, when the *NSA (United States National Security Agency) scandal* was leaked to the public, people started to opt for the services which can provide them a strong privacy for their data. Among the services people opted for, most particularly for Emails, were different plug-ins and extensions for their browsers. Interestingly, among the various plug-ins and extensions that people started to use, there were two main programs that were solely responsible for the complete email security that the people needed. One was S/MIME which we will see later and the other was PGP.

As said, PGP (Pretty Good Privacy), is a popular program that is used to provide confidentiality and authentication services for electronic mail and file storage. It was designed by Phil Zimmermann way back in 1991. He designed it in such a way, that the best cryptographic algorithms such as RSA, Diffie-Hellman key exchange, DSS are used for the public-key encryption (or) asymmetric encryption; CAST-128, 3DES, IDEA are used for symmetric encryption and SHA-1 is used for hashing purposes. PGP software is an open source one and is not dependent on either of the OS (Operating System) or the processor. The application is based on a few commands which are very easy to use.

The following are the services offered by PGP:

- Authentication
- Confidentiality
- Compression
- Email Compatibility
- Segmentation

4.1.1 Authentication:

Authentication basically means something that is used to validate something as true or real. To login into some sites sometimes we give our account name and password that is an authentication verification procedure.

In the email world, checking the authenticity of an email is nothing but to check *whether it actually came from the person it says*. In emails, authentication has to be checked as there are some people who spoof the emails or some spams and sometimes it can cause a lot of inconvenience. The Authentication service in PGP is provided as follows:

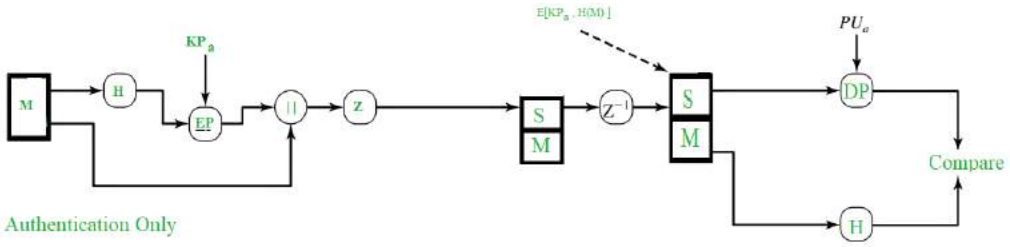


Figure 4.1 Authentication

As shown in the above figure, the Hash Function (H) calculates the Hash Value of the message. For the hashing purpose, SHA-1 is used and it produces a 160-bit output hash value. Then, using the sender's private key (KP_a), it is encrypted and it's called as Digital Signature. The Message is then appended to the signature. All the process happened till now, is sometimes described as *signing the message.* Then the message is compressed to reduce the transmission overhead and is sent over to the receiver.

At the receiver's end, the data is decompressed and the message, signature are obtained. The signature is then decrypted using the sender's public key (PU_a) and the hash value is obtained. The message is again passed to hash function and it's hash value is calculated and obtained.

Both the values, one from signature and another from the recent output of hash function are compared and if both are same, it means that the email is actually sent from a known one and is legit, else it means that it's not a legit one.

2. Confidentiality:
Sometimes we see some packages labelled as 'Confidential', which means that those packages are not meant for all the people and only selected persons can see them. The same applies to the email confidentiality as well. Here, in the email service, only the sender and the receiver should be able to read the message, that means the contents have to be kept secret from every other person, except for those two.

PGP provides that Confidentiality service in the following manner:

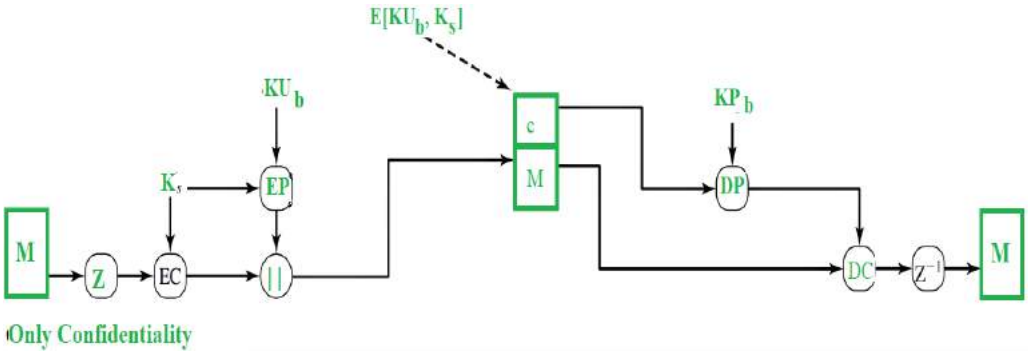


Figure 4.2 Confidentiality

The message is first compressed and a 128-bit session key (K_s), generated by the PGP, is used to encrypt the message through symmetric encryption. Then, the session key (K_s) itself gets encrypted through public key encryption (EP) using receiver's public key(KU_b) . Both the encrypted entities are now concatenated and sent to the receiver.

As you can see, the original message was compressed and then encrypted initially and hence even if anyone could get hold of the traffic, he cannot read the contents as they are not in readable form and they can only read them if they had the session key (K_s). Even though session key is transmitted to the receiver and hence, is in the traffic, it is in encrypted form and only the receiver's private key (KP_b)can be used to decrypt that and thus our message would be completely safe.

At the receiver's end, the encrypted session key is decrypted using receiver's private key (KP_b) and the message is decrypted with the obtained session key. Then, the message is decompressed to obtain the original message (M).

RSA algorithm is used for the public-key encryption and for the symmetric jey encryption, CAST-128(or IDEA or 3DES) is used.

Practically, both the Authentication and Confidentiality services are provided in parallel as follows:

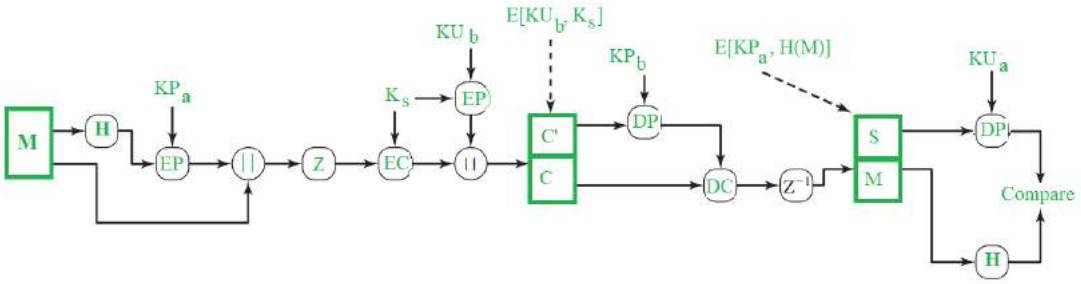


Figure 4.3 Authentication and Confidentiality

Note:
M – Message
H – Hash Function
K_s – A random Session Key created for Symmetric Encryption purpose
DP – Public-Key Decryption Algorithm
EP – Public-Key Encryption Algorithm
DC – Asymmetric Encryption Algorithm
EC – Symmetric Encryption Algorithm
KP_b – A private key of user B used in Public-key encryption process
KP_a – A private key of user A used in Public-key encryption process
PU_a – A public key of user A used in Public-key encryption process
PU_b – A public key of user B used in Public-key encryption process
|| – Concatenation
Z – Compression Function
Z^{-1} – Decompression Function

4.2. Secure /Multipurpose Internet Mail Extensions (S/MIME)

What is S/MIME?

S/MIME is a protocol for the secure exchange of e-mail and attached documents originally developed by RSA Security. Secure/Multipurpose Internet Mail Extensions (S/MIME) adds security to Internet e-mail based on the Simple Mail Transfer Protocol (SMTP) method and adds support for digital signatures and encryption to SMTP mail to support authentication of the sender and privacy of the communication. Note that because HTTP messages can transport MIME data, they can also use S/MIME.

How It Works

S/MIME is an extension of the widely implemented Multipurpose Internet Mail Extensions (MIME) encoding standard, which defines how the body portion of an SMTP message is structured and formatted. S/MIME uses the RSA public key cryptography algorithm along with the Data Encryption Standard (DES) or Rivest-Shamir-Adleman (RSA) encryption algorithm. In an S/MIME message, the MIME body section consists of a message in PKCS #7 format that contains an encrypted form of the MIME body parts. The MIME content type for the encrypted data is application/pkcs7-mime.

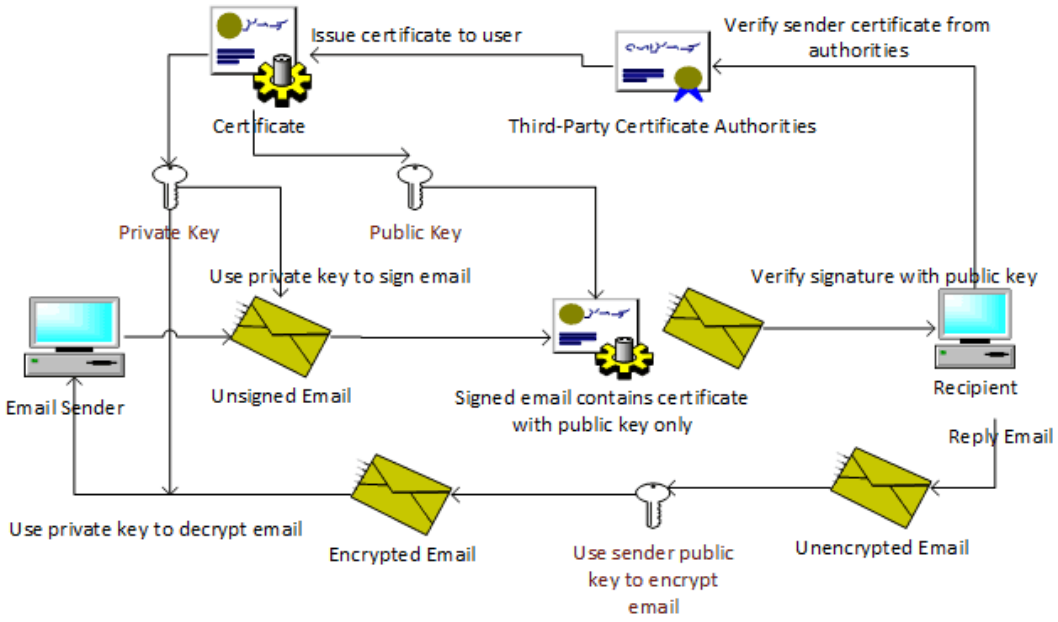


Figure 4.4 S/Mime Structure

Understanding Digital Signatures

Digital signatures are the more commonly used service of S/MIME. As the name suggests, digital signatures are the digital counterpart to the traditional, legal signature on a paper document. As with a legal signature, digital signatures provide the following security capabilities:

- Authentication A signature serves to validate an identity. It verifies the answer to "who are you" by providing a means of differentiating that entity from all others and proving its uniqueness. Because there is no authentication in SMTP e-mail, there is no way to know who actually sent a message. Authentication in a digital signature solves this problem by allowing a recipient to know that a message was sent by the person or organization who claims to have sent the message.

- Nonrepudiation The uniqueness of a signature prevents the owner of the signature from disowning the signature. This capability is called nonrepudiation. Thus, the authentication that a signature provides gives the means to enforce nonrepudiation. The concept of nonrepudiation is most familiar in the context of paper contracts: a signed contract is a legally binding document, and it is impossible to disown an authenticated signature. Digital signatures provide the same function and, increasingly in some areas, are recognized as legally binding, similar to a signature on paper. Because SMTP e-mail does not provide a means of authentication, it cannot provide nonrepudiation. It is easy for a sender to disavow ownership of an SMTP e-mail message.

- Data integrity An additional security service that digital signatures provide is data integrity. Data integrity is a result of the specific operations that make digital signatures possible. With data integrity services, when the recipient of a digitally signed e-mail message

validates the digital signature, the recipient is assured that the e-mail message that is received is, in fact, the same message that was signed and sent, and has not been altered while in transit. Any alteration of the message while in transit after it has been signed invalidates the signature. In this way, digital signatures are able to provide an assurance that signatures on paper cannot, because it is possible for a paper document to be altered after it has been signed.

4.3 Secure Electronic Transaction (SET) Protocol

Secure Electronic Transaction or SET is a system which ensures security and integrity of electronic transactions done using credit cards in a scenario. SET is not some system that enables payment but it is a security protocol applied on those payments. It uses different encryption and hashing techniques to secure payments over internet done through credit cards. SET protocol was supported in development by major organizations like Visa, Mastercard, Microsoft which provided its Secure Transaction Technology (STT) and NetScape which provided technology of Secure Socket Layer (SSL).

SET protocol restricts revealing of credit card details to merchants thus keeping hackers and thieves at bay. SET protocol includes Certification Authorities for making use of standard Digital Certificates like X.509 Certificate.

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

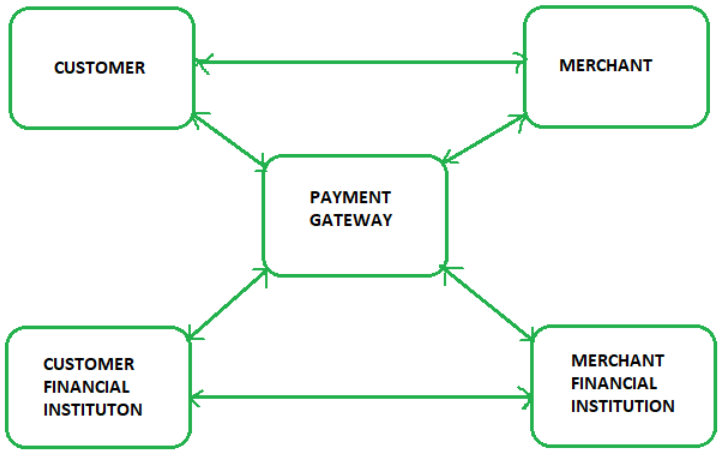


Figure 4.5 SET protocol

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

- It has to provide mutual authentication i.e., customer (or cardholder) authentication by confirming if the customer is intended user or not and merchant authentication.
- It has to keep the PI (Payment Information) and OI (Order Information) confidential by appropriate encryptions.

- It has to be resistive against message modifications i.e., no changes should be allowed in the content being transmitted.
- SET also needs to provide interoperability and make use of best security mechanisms.

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

- Cardholder – customer
- Issuer – customer financial institution
- Merchant
- Acquirer – Merchant financial
- Certificate authority – Authority which follows certain standards and issues certificates (like X.509V3) to all other participants.

SET functionalities:

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

Dual Signature:

The dual signature is a concept introduced with SET, which aims at connecting two information pieces meant for two different receivers:

- Order Information (OI) for merchant
- Payment Information (PI) for bank.

You might think sending them separately is an easy and more secure way, but sending them in a connected form resolves any future dispute possible. Here is the generation of dual signature:

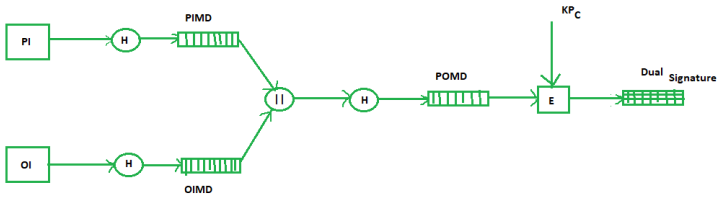


Figure 4.6 Generation of dual signature

Where,

PI stands for payment information
OI stands for order information
PIMD stands for Payment Information Message Digest
OIMD stands for Order Information Message Digest
POMD stands for Payment Order Message Digest
H stands for Hashing
E stands for public key encryption
KPc is customer's private key
|| stands for append operation
Dual signature, DS= E(KPc, [H(H(PI)||H(OI))])

Purchase Request Generation:
The process of purchase request generation requires three inputs:

- Payment Information (PI)
- Dual Signature
- Order Information Message Digest (OIMD)

The purchase request is generated as follows:

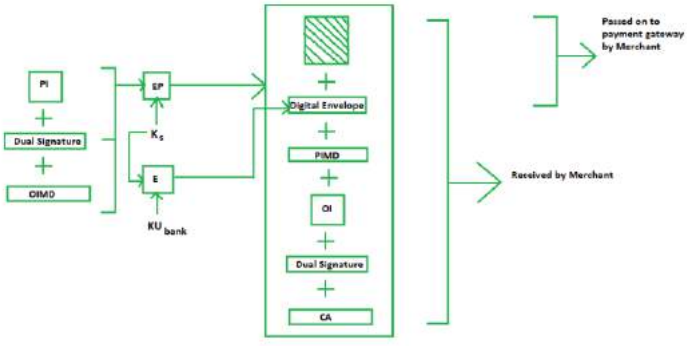


Figure 4.7 Purchase Request Generation

Here,

PI, OIMD, OI all have the same meanings as before.

The new things are :

EP which is symmetric key encryption

Ks is a temporary symmetric key

KUbank is public key of bank

CA is Cardholder or customer Certificate

Digital Envelope = E(KUbank, Ks)

Purchase Request Validation on Merchant Side :
The Merchant verifies by comparing POMD generated through PIMD hashing with POMD generated through decryption of Dual Signature as follows:

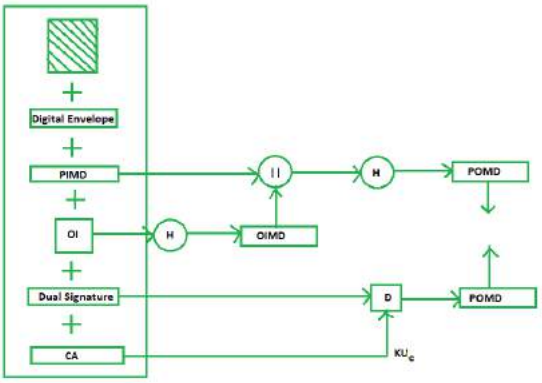


Figure 4.8 POMD generated through decryption of Dual Signature

Since we used Customer private key in encryption here we use KUc which is public key of customer or cardholder for decryption ‘D’.

Payment Authorization and Payment Capture:

Payment authorization as the name suggests is the authorization of payment information by merchant which ensures payment will be received by merchant. Payment capture is the process by which merchant receives payment which includes again generating some request blocks to gateway and payment gateway in turn issues payment to merchant.

4.4. IP security (IPSec)

The IP sec4rity (IPSec) is an Internet Engineering Task Force (IETF) standard suite of protocols between 2 communication points across the IP network that provide data authentication, integrity, and confidentiality. It also defines the encrypted, decrypted and authenticated packets. The protocols needed for secure key exchange and key management are defined in it.

Uses of IP Security –
IPsec can be used to do the following things:

- To encrypt application layer data.
- To provide security for routers sending routing data across the public internet.
- To provide authentication without encryption, like to authenticate that the data originates from a known sender.
- To protect network data by setting up circuits using IPsec tunneling in which all data is being sent between the two endpoints is encrypted, as with a Virtual Private Network(VPN) connection.

Components of IP Security –
It has the following components:

- Encapsulating Security Payload (ESP): It provides data integrity, encryption, authentication and anti-replay. It also provides authentication for payload.
- Authentication Header (AH) : It also provides data integrity, authentication and anti-replay and it does not provide encryption. The anti-replay protection, protects against unauthorized transmission of packets. It does not protect data's confidentiality.

Figure 4.9 Authentication Header

- Internet Key Exchange (IKE)

It is a network security protocol designed to dynamically exchange encryption keys and find a way over Security Association (SA) between 2 devices. The Security Association (SA) establishes shared security attributes between 2 network entities to support secure communication. The Key Management Protocol (ISAKMP) and Internet Security Association which provides a framework for authentication and key exchange. ISAKMP tells how the setup of the Security Associations (SAs) and how direct connections between two hosts that are using IPsec.

Internet Key Exchange (IKE) provides message content protection and also an open frame for implementing standard algorithms such as SHA and MD5. The algorithm's IP sec users produces a unique identifier for each packet. This identifier then allows a device to determine whether a packet has been correct or not. Packets which are not authorized are discarded and not given to receiver.

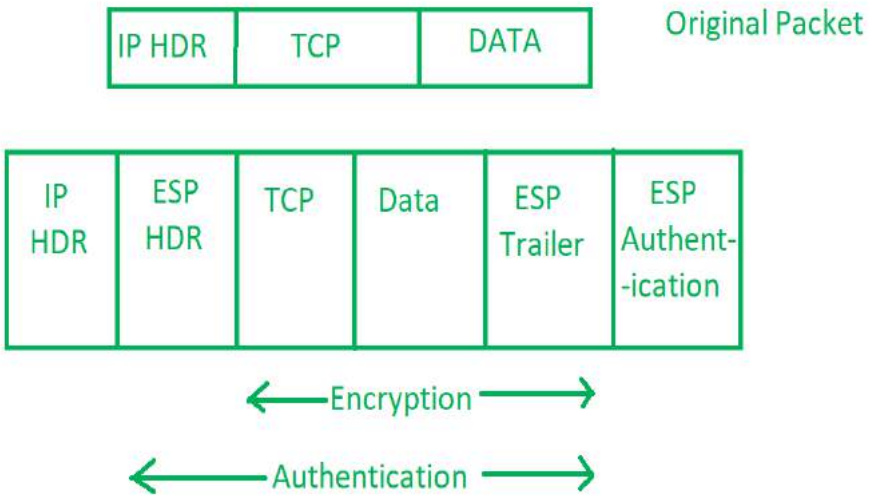


Figure 4.10 Internet Key Exchange

Working of IP Security –

- The host checks if the packet should be transmitted using IPsec or not. These packet traffic triggers the security policy for themselves. This is done when the system sending the packet apply an appropriate encryption. The incoming packets are also checked by the host that they are encrypted properly or not.
- Then the IKE Phase 1 starts in which the 2 hosts(using IPsec) authenticate themselves to each other to start a secure channel. It has 2 modes. The Main mode which provides the greater security and the Aggressive mode which enables the host to establish an IPsec circuit more quickly.
- The channel created in the last step is then used to securely negotiate the way the IP circuit will encrypt data accross the IP circuit.
- Now, the IKE Phase 2 is conducted over the secure channel in which the two hosts negotiate the type of cryptographic algorithms to use on the session and agreeing on secret keying material to be used with those algorithms.
- Then the data is exchanged across the newly created IPsec encrypted tunnel. These packets are encrypted and decrypted by the hosts using IPsec SAs.
- When the communication between the hosts is completed or the session times out then the IPsec tunnel is terminated by discarding the keys by both the hosts.

IPSec Architecture

IPSec (IP Security) architecture uses two protocols to secure the traffic or data flow. These protocols are ESP (Encapsulation Security Payload) and AH (Authentication Header). IPSec Architecture include protocols, algorithms, DOI, and Key Management. All these components are very important in order to provide the three main services:

- Confidentiality
- Authentication
- Integrity

IP Security Architecture:

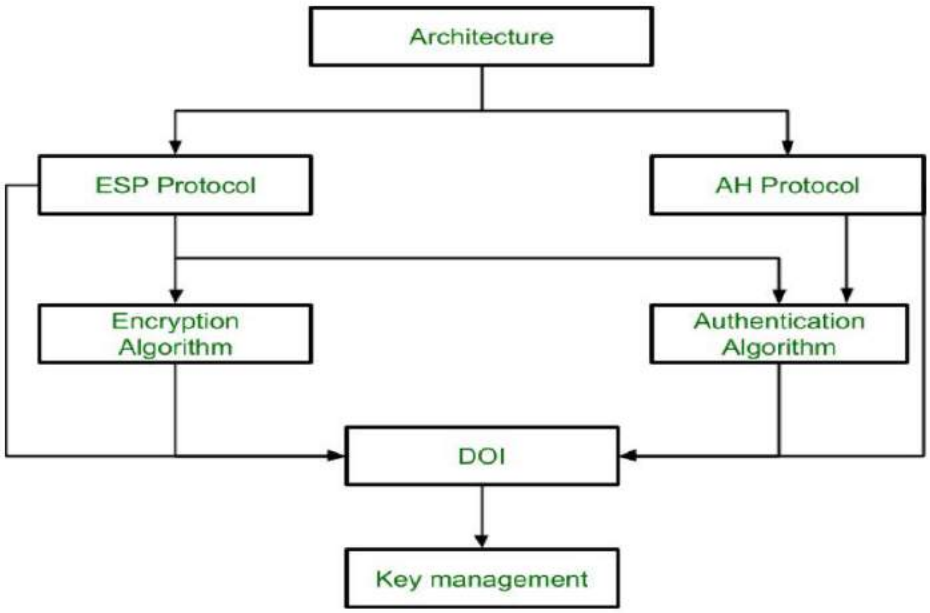


Figure 4.11 IP Security Architecture

- Architecture:
  Architecture or IP Security Architecture covers the general concepts, definitions, protocols, algorithms and security requirements of IP Security technology.
- ESP Protocol:
  ESP (Encapsulation Security Payload) provide the confidentiality service. Encapsulation Security Payload is implemented in either two ways:
  - ESP with optional Authentication.
  - ESP with Authentication.

Packet Format:

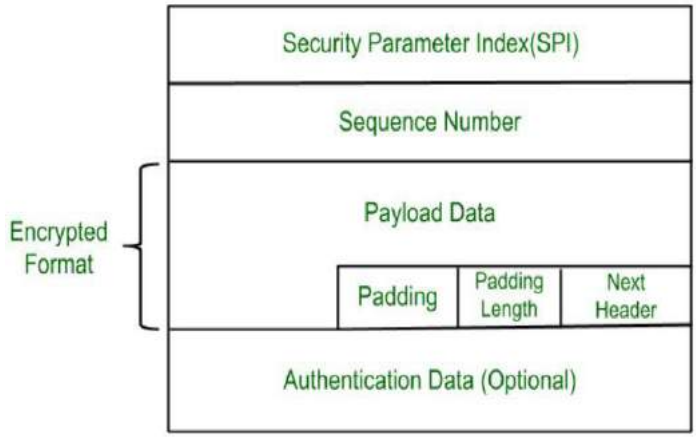


Figure 4.12 Packet Format

- Security Parameter Index (SPI):
  This parameter is used in Security Association. It is used to give a unique number to the connection build between Client and Server.
- Sequence Number:
  Unique Sequence number are allotted to every packet so that at the receiver side packets can be arranged properly.
- Payload Data:
  Payload data means the actual data or the actual message. The Payload data is in encrypted format to achieve confidentiality.
- Padding:
  Extra bits or space added to the original message in order to ensure confidentiality. Padding length is the size of the added bits or space in the original message.
- Next Header:
  Next header means the next payload or next actual data.
- Authentication Data
  This field is optional in ESP protocol packet format.

- Encryption algorithm:
  Encryption algorithm is the document that describes various encryption algorithm used for Encapsulation Security Payload.
- AH Protocol:
  AH (Authentication Header) Protocol provides both Authentication and Integrity service. Authentication Header is implemented in one way only: Authentication along with Integrity.

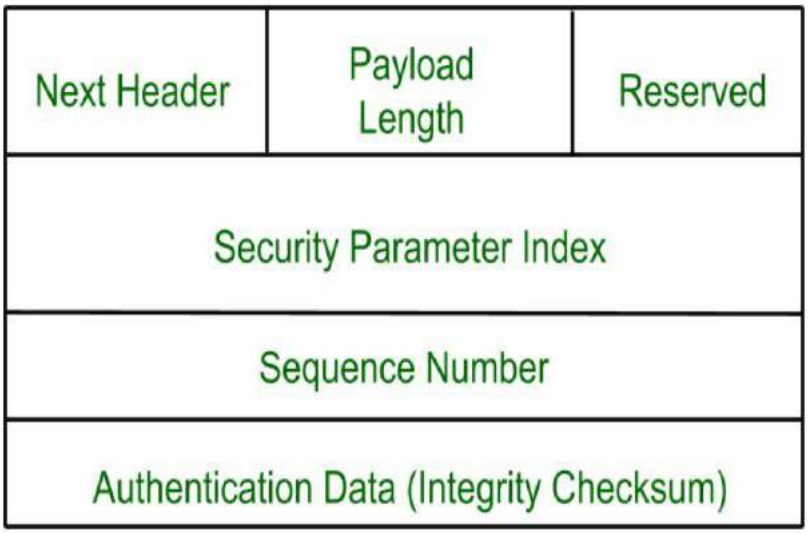


Figure 4.13 AH Protocol

Authentication Header covers the packet format and general issue related to the use of AH for packet authentication and integrity.

- Authentication Algorithm:
  Authentication Algorithm contains the set of the documents that describe authentication algorithm used for AH and for the authentication option of ESP.
- DOI (Domain of Interpretation):
  DOI is the identifier which support both AH and ESP protocols. It contains values needed for documentation related to each other.
- Key Management:
  Key Management contains the document that describes how the keys are exchanged between sender and receiver.

4.5 Encapsulating Security Payload (ESP)

Encapsulating Security Payload (ESP) is a member of the Internet Protocol Security (IPsec) set of protocols that encrypt and authenticate the packets of data between computers using a Virtual Private Network (VPN). The focus and layer on which ESP operates makes it possible for VPNs to function securely.

Being one of the most popular tools used in network security, Encapsulating Security Payload (abbreviated as ESP) offers the help we need in keeping the integrity, authenticity and confidentiality of the information we send across networks. Keep reading to learn more!

With the technological advancements, the way we conduct our business processes has changed immensely. Now, we heavily rely on the internet technologies and transfer massive amounts of data daily. For this data traffic, we often employ wireless and wired networks. As a result, network security and necessary cybersecurity measures gain importance each day.

Being one of the most popular tools used in network security, Encapsulating Security Payload (abbreviated as ESP) offers the help we need in keeping the integrity, authenticity and confidentiality of the information we send across networks. In this article, we will take a closer look at what Encapsulating Security Payload is. Keep reading to learn more.

What is Encapsulating Security Payload?

Encapsulating Security Payload (abbr. ESP) is a protocol within the scope of the IPSec.

The information traffic on a network is provided with packets of data. In other words, when you want to send or receive a data through a network, it is turned into packets of information so that it can travel within the network. Similar to the data packages, payload is also sent through the network and it contains the ‘actual’ information, the intended message.

The Encapsulating Security Payload aims to offer necessary security measures for these packets of data and/or payloads. With the help of Encapsulating Security Payload, confidentiality, integrity and authentication of payloads and data packets in IPv4 and IPv6 networks.

How does the Encapsulating Security Payload work?

Also known as a transport layer security protocol, the Encapsulating Security Payload is able to function with both the IPv6 and IPv4 protocols. The way ESP operates is pretty straightforward: It is inserted between the Internet Protocol/IP header and upper layer protocols such as UDP, ICMP or TCP. In this position, the ESP takes the form of a header.

How can the Encapsulating Security Payload be used?

Although the Encapsulating Security Payload offers many benefits, it can be applied in only two ways: Tunnel mode and transport mode.

In the tunnel mode, a new IP header is created and used as the outermost IP header. It is followed by the Encapsulating Security Payload Header and original datagram. Tunnel mode is a must for the gateways.

In the transportation mode, the IP header is neither authenticated nor encrypted. As a result, your addressing information can potentially be leaked during the datagram transit. Transport mode often uses less processing, that is why most hosts prefer Encapsulating Security Payload in transport mode.

What are the benefits of the Encapsulating Security Payload?

The Encapsulating Security Payload offers all the functions of the Authentication Header, which are anti-replay protection, authentication and data integrity. On the other hand, the ESP differs from the Authentication Header in terms of data confidentiality: the ESP can provide data confidentiality while the Authentication Header cannot.

Moreover, the Encapsulating Security Protocol Payload aims to provide various services including but not limited to:

- Maintaining the confidentiality of datagrams with encryption
- Using security gateways to limit the traffic flow confidentiality
- Authenticating the origin of data using a public key encryption
- Providing antireplay services with the help of the sequence number mechanism given by the Authentication Header

In business environments, we use network technologies very often. They allow us to share resources and files, set communication protocols and such. As much as they streamline and accelerate our business processes, they can also pose a serious vulnerability for our cyber security. An intruder or a hacker can infiltrate into our networks, steal our valuable information or lock us out of our systems. That is why network security is one of the most important practices in cybersecurity.

Most organizations rely on firewalls for their network security needs. A firewall can be defined as a network security system that allows the cybersecurity professionals to monitor and control the network traffic. In other words, a firewall sets the boundary between the internal and external network. There are two main types of firewalls:

- Network-based firewalls: They are often positioned on the LANs, intranets or WANs of the gateway computers.

- Host-based firewalls: They are implemented on the network host itself in order to protect the entire network traffic. Host-based firewalls can be a part of the operating system or an agent application in order to offer an additional layer of security.

What is stateful inspection?

The term stateful inspection (also known as the dynamic packet filtering) refers to a distinguished firewall technology. It aims to monitor the active connections on a network. Moreover, the process of stateful inspection determines which network packets should be allowed through the firewall by utilizing the information regarding active connections.

Stateful inspection keeps track of each connection and constantly checks if they are valid. That is why it offers a better protection than its predecessors.

In a firewall where the stateful inspection is implemented, the network administrator can customise the parameters in order to meet the unique needs of the organization.

What is the benefit of implementing stateful inspection?

Before stateful inspection has become mainstream, similar technology called static packet filtering was in use. This older alternative only checks the headers of the packets in order to determine whether they should be allowed through the firewall. As a result, a hacker can simply indicate “reply” in the header in order to extract information from the network. On the contrary, stateful inspection aims to carry out a more sophisticated investigation. That is why it analyses the application layer of the packets. A dynamic packet filter like stateful inspection can offer a better security posture for networks through recording the session information like port numbers or IP addresses.

In other words, stateful inspection is better at keeping the intruders away from your network since it uses a more refined technology.

4.6 Internet key Exchange-

Internet Key Exchange (IKE) is a key management protocol standard used in conjunction with the Internet Protocol Security (IPSec) standard protocol. It provides security for virtual private networks' (VPNs) negotiations and network access to random hosts. It can also be described as a method for exchanging keys for encryption and authentication over an unsecured medium, such as the Internet.

IKE is a hybrid protocol based on:

- ISAKMP (RFC2408): Internet Security Association and Key Management Protocols are used for negotiation and establishment of security associations. This protocol establishes a secure connection between two IPSec peers.
- Oakley (RFC2412): This protocol is used for key agreement or key exchange. Oakley defines the mechanism that is used for key exchange over an IKE session. The default algorithm for key exchange used by this protocol is the Diffie-Hellman algorithm.
- SKEME: This protocol is another version for key exchange.

IKE enhances IPsec by providing additional features along with flexibility. IPsec, however, can be configured without IKE.

IKE has many benefits. It eliminates the need to manually specify all the IPSec security parameters at both peers. It allows the user to specify a particular lifetime for the IPsec security association. Furthermore, encryption can be changed during IPsec sessions. Moreover, it permits certification authority. Finally, it allows dynamic authentication of peers.

The IKE works in two steps. The first step establishes an authenticated communication channel between the peers, by using algorithms like the Diffie-Hellman key exchange, which generates a shared key to further encrypt IKE communications. The communication channel formed as a result of the algorithm is a bi-directional channel. The authentication of the channel is achieved by using a shared key, signatures, or public key encryption.

There are two modes of operation for the first step: main mode, which is utilized to protect the identity of the peers, and aggressive mode, which is used when the security of the identity of the peers is not an important issue. During the second step, the peers use the secure communication channel to set up security negotiations on behalf of other services like IPSec. These negotiation procedures give rise to two unidirectional channels of which one is inbound and the other outbound. The mode of operation for the second step is the Quick mode.

IKE provides three different methods for peer authentication: authentication using a pre-shared secret, authentication using RSA encrypted nonces, and authentication using RSA signatures. IKE uses the HMAC functions to guarantee the integrity of an IKE session. When an IKE session lifetime expires, a new Diffie-Hellman exchange is performed and the IKE SA is re-established.

4.7. Firewalls

Introduction of Firewall

Firewall is a network device that isolates organization's internal network from larger outside network/Internet. It can be a hardware, software, or combined system that prevents unauthorized access to or from internal network.

A firewall is a network security device, either hardware or software-based, which monitors all incoming and outgoing traffic and based on a defined set of security rules it accepts, rejects or drops that specific traffic.

Accept : allow the traffic
Reject : block the traffic but reply with an “unreachable error”
Drop : block the traffic with no reply

A firewall establishes a barrier between secured internal networks and outside untrusted network, such as the Internet.

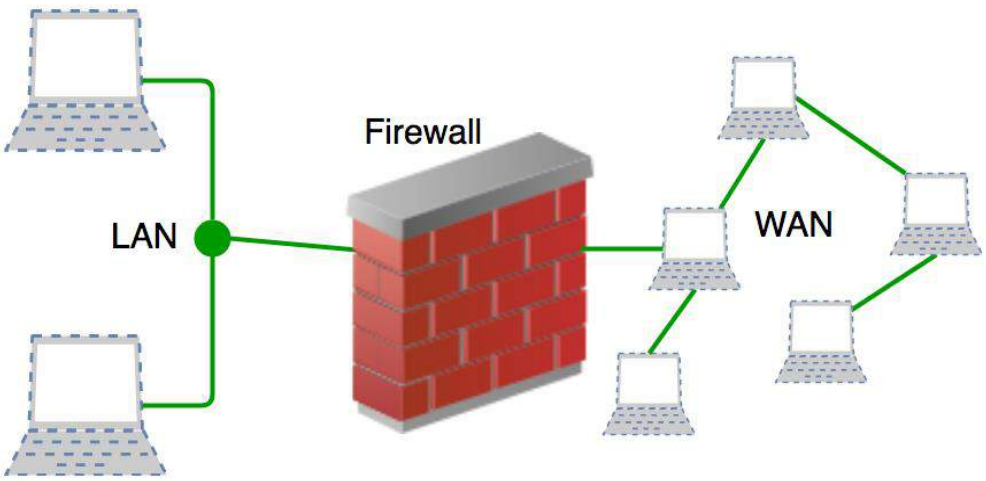


Figure 4.14 Basic Structure of Firewall

History and Need for Firewall

Before Firewalls, network security was performed by Access Control Lists (ACLs) residing on routers. ACLs are rules that determine whether network access should be granted or denied to specific IP address.

But ACLs cannot determine the nature of the packet it is blocking. Also, ACL alone does not have the capacity to keep threats out of the network. Hence, the Firewall was introduced.

Connectivity to the Internet is no longer optional for organizations. However, accessing the Internet provides benefits to the organization; it also enables the outside world to interact with the internal network of the organization. This creates a threat to the organization. In order to secure the internal network from unauthorized traffic, we need a Firewall.

How Firewall Works

Firewall match the network traffic against the rule set defined in its table. Once the rule is matched, associate action is applied to the network traffic. For example, Rules are defined as any employee from HR department cannot access the data from code server and at the same time another rule is defined like system administrator can access the data from both HR and technical department. Rules can be defined on the firewall based on the necessity and security policies of the organization.

From the perspective of a server, network traffic can be either outgoing or incoming. Firewall maintains a distinct set of rules for both the cases. Mostly the outgoing traffic, originated from the server itself, allowed to pass. Still, setting a rule on outgoing traffic is always better in order to achieve more security and prevent unwanted communication. Incoming traffic is treated differently. Most traffic which reaches on the firewall is one of these three major Transport Layer protocols- TCP, UDP or ICMP. All these types have a source address and destination address. Also, TCP and UDP have port numbers. ICMP uses *type code* instead of port number which identifies purpose of that packet.

Default policy: It is very difficult to explicitly cover every possible rule on the firewall. For this reason, the firewall must always have a default policy. Default policy only consists of action (accept, reject or drop).

Suppose no rule is defined about SSH connection to the server on the firewall. So, it will follow the default policy. If default policy on the firewall is set to *accept*, then any computer outside of your office can establish an SSH connection to the server. Therefore, setting default policy as *drop* (or reject) is always a good practice.

Generation of Firewall

Firewalls can be categorized based on its generation.

- First Generation- Packet Filtering Firewall: Packet filtering firewall is used to control network access by monitoring outgoing and incoming packet and allowing them to pass or stop based on source and destination IP address, protocols and ports. It analyses traffic at the transport protocol layer (but mainly uses first 3 layers). Packet firewalls treat each packet in isolation. They have no ability to tell whether a packet is part of an existing stream of traffic. Only It can allow or deny the packets based on unique packet headers.
- Packet filtering firewall maintains a filtering table which decides whether the packet will be forwarded or discarded. From the given filtering table, the packets will be Filtered according to following rules:

	Source IP	Dest. IP	Source Port	Dest. Port	Action
1	192.168.21.0	--	--	--	deny
2	--	--	--	23	deny
3	--	192.168.21.3	--	--	deny
4	--	192.168.21.0	--	>1023	Allow

- Figure 4.15 Sample packet Filter Firewall Rule
  - Incoming packets from network 192.168.21.0 are blocked.
  - Incoming packets destined for internal TELNET server (port 23) are blocked.
  - Incoming packets destined for host 192.168.21.3 are blocked.
  - All well-known services to the network 192.168.21.0 are allowed.
- Second Generation- Stateful Inspection Firewall: Stateful firewalls (performs Stateful Packet Inspection) are able to determine the connection state of packet, unlike Packet filtering firewall, which makes it more efficient. It keeps track of the state of networks connection travelling across it, such as TCP streams. So the filtering decisions would not

only be based on defined rules, but also on packet's history in the state table.

- Third Generation- Application Layer Firewall: Application layer firewall can inspect and filter the packets on any OSI layer, up to the application layer. It has the ability to block specific content, also recognize when certain application and protocols (like HTTP, FTP) are being misused.
- In other words, Application layer firewalls are hosts that run proxy servers. A proxy firewall prevents the direct connection between either side of the firewall, each packet has to pass through the proxy. It can allow or block the traffic based on predefined rules.
- Note: Application layer firewalls can also be used as Network Address Translator (NAT).
- Next Generation Firewalls (NGFW) : Next Generation Firewalls are being deployed these days to stop modern security breaches like advance malware attacks and application-layer attacks. NGFW consists of Deep Packet Inspection, Application Inspection, SSL/SSH inspection and many functionalities to protect the network from these modern threats.

Firewalls are generally of two types: *Host-based* and *Network-based.*

- Host- based Firewalls: Host-based firewall is installed on each network node which controls each incoming and outgoing packet. It is a software application or suite of applications, comes as a part of the operating system. Host-based firewalls are needed because network firewalls cannot provide protection inside a trusted network. Host firewall protects each host from attacks and unauthorized access.
- Network-based Firewalls: Network firewall function on network level. In other words, these firewalls filter all incoming and outgoing traffic across the network. It protects the internal network by filtering the traffic using rules defined on the firewall. A Network firewall might have two or more network interface cards (NICs). A network-based firewall is usually a dedicated system with proprietary software installed.

Both types of firewall have their own advantages.

Firewall is categorized into three basic types

- Packet filter (Stateless & Stateful)
- Application-level gateway
- Circuit-level gateway

These three categories, however, are not mutually exclusive. Modern firewalls have a mix of abilities that may place them in more than one of the three categories.

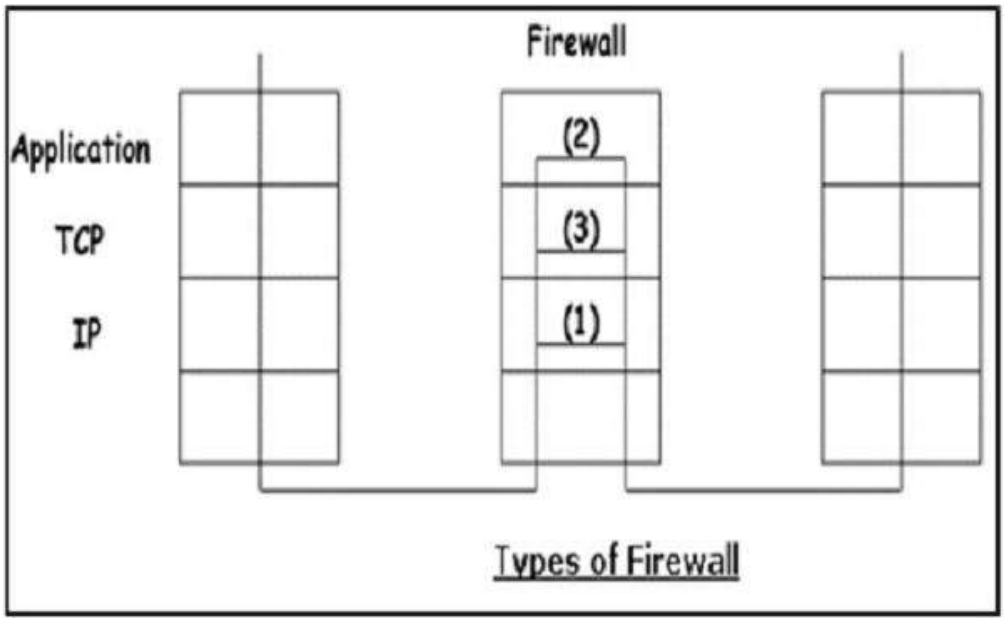


Figure 4.16 Types of firewall

Stateless & Stateful Packet Filtering Firewall

In this type of firewall deployment, the internal network is connected to the external network/Internet via a router firewall. The firewall inspects and filters data packet-by-packet.

Packet-filtering firewalls allow or block the packets mostly based on criteria such as source and/or destination IP addresses, protocol, source and/or destination port numbers, and various other parameters within the IP header.

The decision can be based on factors other than IP header fields such as ICMP message type, TCP SYN and ACK bits, etc.

Packet filter rule has two parts −

- Selection criteria − It is a used as a condition and pattern matching for decision making.
- Action field − This part specifies action to be taken if an IP packet meets the selection criteria. The action could be either block (deny) or permit (allow) the packet across the firewall.

Packet filtering is generally accomplished by configuring Access Control Lists (ACL) on routers or switches. ACL is a table of packet filter rules.

As traffic enters or exits an interface, firewall applies ACLs from top to bottom to each incoming packet, finds matching criteria and either permits or denies the individual packets.

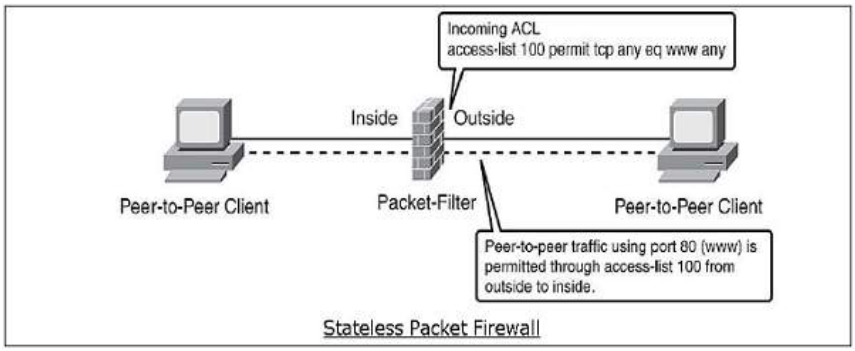


Figure 4.17 Stateless firewall

Stateless firewall is a kind of a rigid tool. It looks at packet and allows it if its meets the criteria even if it is not part of any established ongoing communication.

Hence, such firewalls are replaced by stateful firewalls in modern networks. This type of firewalls offer a more in-depth inspection method over the only ACL based packet inspection methods of stateless firewalls.

Stateful firewall monitors the connection setup and teardown process to keep a check on connections at the TCP/IP level. This allows them to keep track of connections state and determine which hosts have open, authorized connections at any given point in time.

They reference the rule base only when a new connection is requested. Packets belonging to existing connections are compared to the firewall's state table of open connections, and decision to allow or block is taken. This process saves time and provides added security as well. No packet is allowed to trespass the firewall unless it belongs to already established connection. It can timeout inactive connections at firewall after which it no longer admit packets for that connection.

Application Gateways

An application-level gateway acts as a relay node for the application-level traffic. They intercept incoming and outgoing packets, run proxies that copy and forward information across the gateway, and function as a *proxy server*, preventing any direct connection between a trusted server or client and an untrusted host.

The proxies are application specific. They can filter packets at the application layer of the OSI model.

Application-specific Proxies

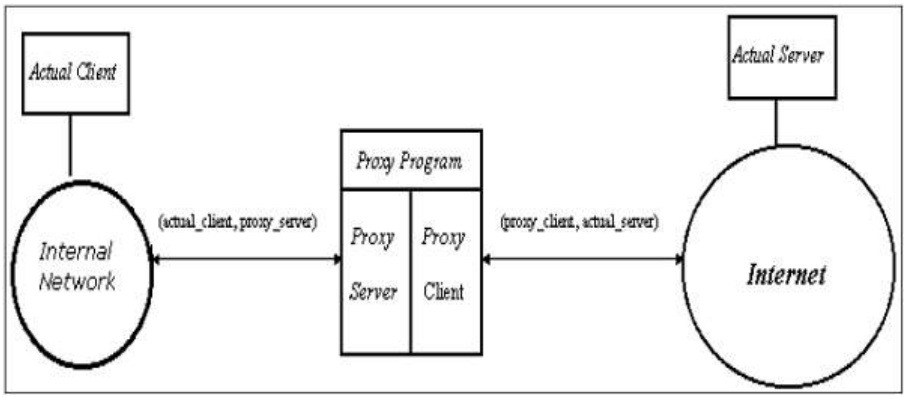


Figure 4.18 Application-specific Proxies

An application-specific proxy accepts packets generated by only specified application for which they are designed to copy, forward, and filter. For example, only a Telnet proxy can copy, forward, and filter Telnet traffic.

If a network relies only on an application-level gateway, incoming and outgoing packets cannot access services that have no proxies configured. For example, if a gateway runs FTP and Telnet proxies, only packets generated by these services can pass through the firewall. All other services are blocked.

Application-level Filtering

An application-level proxy gateway, examines and filters individual packets, rather than simply copying them and blindly forwarding them across the gateway. Application-specific proxies check each packet that passes through the gateway, verifying the contents of the packet up through the application layer. These proxies can filter particular kinds of commands or information in the application protocols.

Application gateways can restrict specific actions from being performed. For example, the gateway could be configured to prevent users from performing the 'FTP put' command. This can prevent modification of the information stored on the server by an attacker.

Transparent

Although application-level gateways can be transparent, many implementations require user authentication before users can access an untrusted network, a process that reduces true transparency. Authentication may be different if the user is from the internal network or from the Internet. For an internal network, a simple list of IP addresses can be allowed to connect to external applications. But from the Internet side a strong authentication should be implemented.

An application gateway actually relays TCP segments between the two TCP connections in the two directions (Client ↔ Proxy ↔ Server).

For outbound packets, the gateway may replace the source IP address by its own IP address. The process is referred to as Network Address Translation (NAT). It ensures that internal IP addresses are not exposed to the Internet.

Circuit-Level Gateway

The circuit-level gateway is an intermediate solution between the packet filter and the application gateway. It runs at the transport layer and hence can act as proxy for any application.

Similar to an application gateway, the circuit-level gateway also does not permit an end-to-end TCP connection across the gateway. It sets up two TCP connections and relays the TCP segments from one network to the other. But, it does not examine the application data like application gateway. Hence, sometime it is called as 'Pipe Proxy'.

4.8 Hybrids Systems: In an attempt combine the security of the application layer gateways with the flexibility and speed of packet filtering; some vendors have created systems that use the principle of both.

In some of these systems, new connections must be authenticated and approved at the application layer. Once this has been done, the remainder of the connection is passed down to the session layer, where packet filters watch the connection to ensure that only packets that are part of an on-going (already authenticated and approved) conversation are-being passed.

Other possibilities include using both packet filtering and application layer proxies. The benefits here include providing a measure of protection against the machines that provide services to the internet (such as a public web server), as well as provide the security of an application layer gateway to the internal network. Additionally, using this method, an attacker, in order to get to services on the internal network, will have to break through the access router, the bastion host, and the choke router.

4.9 Important Aspects of Effective Firewalls

Regardless of which security design logic or packet screening method is chosen, two important aspects of the firewall's implementation can determine whether or not a firewall solution will be effective:

- First, the device or host system on which the firewall solution resides must be secure. If the system can be compromised, then the firewall can also be compromised. If the firewalls you choose is based on a well-known network operating system, make sure the operating system is fully patched and all security updates have been applied. .
- Second, for a firewall to be effective, all traffic to and from your network must pass through it. If a firewall can be physically or logically bypassed, there is no guarantee that the trusted network is safe. The architecture used for the firewall solution is very important.

Since firewall solutions can be configured using a single system or multiple systems, the architecture used to implement the solution can be simple or complex. When deciding on a specific architecture keep in mind that the most effective firewall solutions are implemented to all network traffic passes through them. This implementation characteristic is evident in the following commonly identified firewall architectures.

REFERENCES

1. Willliam Stallings," Network System Essentials "-4th Edition Copyright © 2011 Pearson education, Inc., publishing as [Prentice Hall,
2. Atul Khahate, "Cryptography and network security",3rd Edition, Copyright © 2013 TMH Publishing
3. Kuldeep Singh Kohar", Network Security", revised reprint 2011.Vayu Education of India, New Delhi.Hall, 1983.

Review Questions:

Part A

1. List the services offered by PGP
2. Define is S/MIME
3. Explain the requirements in SET protocol
4. List the components of IP Security
5. Draw the IP Security Architecture
6. What is Encapsulating Security Payload?
7. How does the Encapsulating Security Payload work?
8. How can the Encapsulating Security Payload be used?
9. What is stateful inspection?
10. What is the benefit of implementing stateful inspection?
11. Define Internet Key Exchange
12. Explain different methods for peer authentication of IKE.
13. Explain the Need for Firewall
14. Define Firewall
15. What are the categories of Firewall

Part B

1. Demonstrate SECURE ELECTRONIC TRANSACTION (SET) with suitable example
2. Illustrate the steps involved in working of PGP
3. Explain in detail about Multipurpose Internet Mail Extensions
4. Organise the architecture of IP Security Overview and explain
5. Distinguish different types of Firewalls

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE
www.sathyabama.ac.in

SCHOOL OF COMPUTING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UNIT - 5

SCS1316 - NETWORK SECURITY

UNIT- V COMPUTER SYSTEM SECURITY

Syllabus:

Malicious Software- Types-Backdoor-Worms-Logic bomb -Trojan Horses-Viruses-Classifications-Virus Kits-Email Viruses-Antivirus Approach-Distributed Denial of Service Attacks-Counter Measures-Intrusion Detection System (IDS), Network Based IDS-Host based IDS- Steps involved in deploying IDS

5.1 Malicious Software: The most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems. Such threats are referred to as malicious software, or malware. In this context, we are concerned with threats to application programs as well as utility programs, such as editors and compilers, and kernel-level programs.

Malicious software is software that is intentionally included or inserted in a system for a harmful purpose.

Malicious software can be divided into two categories: those that need a host program, and those that are independent.

- The former, referred to as parasitic, are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples.
- Independent malware is a self-contained program that can be scheduled and run by the operating system. Worms and bot programs are examples.

5.2 Backdoor:

A backdoor, also known as a trapdoor, is a secret entry point into a program that allows someone who is aware of the backdoor to gain access without going through the usual security access procedures. Programmers have used backdoors legitimately for many years to debug and test programs; such a backdoor is called a maintenance hook. This usually is done when the programmer is developing an application that has an authentication procedure, or a long setup, requiring the user to enter many different values to run the application. To debug the program, the developer may wish to gain special privileges or to avoid all the necessary setup and authentication. Backdoors become threats when unscrupulous programmers use them to gain unauthorized access. It is difficult to implement operating system controls for backdoors. Security measures must focus on the program development and software update activities.

5.3 Logic Bomb:

One of the oldest types of program threat, predating viruses and worms, is the logic bomb. The logic bomb is code embedded in some legitimate program that is set to “explode” when certain conditions are met. Examples of conditions that can be used as triggers for a logic bomb are the presence or absence of certain files, a particular day of the week or date, or a particular user running the application. Once triggered, a bomb may alter or delete data or entire files, cause a machine halt, or do some other damage.

Table 5.1 Terminology of Malicious Programs

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 a 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 heterogenous collection or 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 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.

5.4 Trojan Horses:

A Trojan horse is a useful, or apparently useful, program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function. Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly. For example, to gain access to the files of another user on a shared system, a user could create a Trojan horse program that, when executed, changes the invoking user's file permissions so that the files are readable by any user.

Another common motivation for the Trojan horse is data destruction. The program appears to be performing a useful function (e.g., a calculator program), but it may also be quietly deleting the user's files.

Trojan horses fit into one of three models:

- Continuing to perform the function of the original program and additionally performing a separate malicious activity
- Continuing to perform the function of the original program but modifying the function to perform malicious
- Performing a malicious function that completely replaces the function of the original program.

5.5 Viruses:

A computer virus is a piece of software that can "infect" other programs by modifying them; the modification includes injecting the original program with a routine to make copies of the virus program, which can then go on to infect other programs. A virus can do anything that other programs do. The difference is that a virus attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs that is allowed by the privileges of the current user. An unconventional acronym of virus in IT industry is Vital Information Resources Under Siege.

A computer virus has three parts

- Infection mechanism: The means by which a virus spreads, enabling it to replicate. The mechanism is also referred to as the infection vector.
- Trigger: The event or condition that determines when the payload is activated or delivered.
- Payload: What the virus does, besides spreading. The payload may involve damage or may involve benign but noticeable activity.

During its lifetime, a typical virus goes through the following four phases:

- Dormant phase: The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.
- Propagation phase: The virus places a copy of itself into other programs or into certain system areas on the disk. The copy may not be identical to the propagating version;

viruses often morph to evade detection. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase. • Triggering phase: The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself. • Execution phase: The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files. Most viruses carry out their work in a manner that is specific to a particular operating system and, in some cases, specific to a particular hardware platform. Thus, they are designed to take advantage of the details and weaknesses of particular systems.

5.6 Viruses Classification

There has been a continuous arms race between virus writers and writers of antivirus software since viruses first appeared. As effective countermeasures are developed for existing types of viruses, newer types are developed. There is no simple or universally agreed upon classification scheme for viruses. In this section, we classify viruses along two orthogonal axes: the type of target the virus tries to infect and the method the virus uses to conceal itself from detection by users and antivirus software. A virus classification by target includes the following categories:

- Boot sector infector: Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
- File infector: Infects files that the operating system or shell consider to be executable.
- Macro virus: Infects files with macro code that is interpreted by an application.

A virus classification by concealment strategy includes the following categories:

- **Encrypted virus:** A typical approach is as follows. A portion of the virus creates a random encryption key and encrypts the remainder of the virus. The key is stored with the virus. When an infected program is invoked, the virus uses the stored random key to decrypt the virus. When the virus replicates, a different random key is selected. Because the bulk of the virus is encrypted with a different key for each instance, there is no constant bit pattern to observe.
- **Stealth virus:** A form of virus explicitly designed to hide itself from detection by antivirus software. Thus, the entire virus, not just a payload is hidden.
- **Polymorphic virus:** A virus that mutates with every infection, making detection by the "signature" of the virus impossible.
- **Metamorphic virus:** As with a polymorphic virus, a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration, increasing the difficulty of detection. Metamorphic viruses may change their behaviour as well as their appearance.

5.7 Virus Kits

Another weapon in the virus writers' armory is the virus-creation toolkit. Such a toolkit enables a relative novice to quickly create a number of different viruses. Although viruses

created with toolkits tend to be less sophisticated than viruses designed from scratch, the sheer number of new viruses that can be generated using a toolkit creates a problem for antivirus schemes.

5.8 E-Mail Viruses

A more recent development in malicious software is the e-mail virus. The first rapidly spreading e-mail viruses, such as Melissa, made use of a Microsoft Word macro embedded in an attachment. If the recipient opens the e-mail attachment, the Word macro is activated. Then

- The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.
- The virus does local damage on the user's system.

In 1999, a more powerful version of the e-mail virus appeared. This newer version can be activated merely by opening an e-mail that contains the virus rather than opening an attachment. The virus uses the Visual Basic scripting language supported by the e-mail package.

It arrives via e-mail and uses e-mail software features to replicate itself across the Internet. The virus propagates itself as soon as it is activated (either by opening an e-mail attachment or by opening the e-mail) to all of the e-mail addresses known to the infected host. This makes it very difficult for antivirus software to respond before much damage is done. Ultimately, a greater degree of security must be built into Internet utility and application software on PCs to counter the growing threat.

5.9 Antivirus Approaches

The ideal solution to the threat of viruses is prevention: Do not allow a virus to get into the system in the first place, or block the ability of a virus to modify any files containing executable code or macros. This goal is, in general, impossible to achieve, although prevention can reduce the number of successful viral attacks. The next best approach is to be able to do the following:

• Detection: Once the infection has occurred, determine that it has occurred and locate the virus.

• Identification: Once detection has been achieved, identify the specific virus that has infected a program.

• Removal: Once the specific virus has been identified, remove all traces of the virus from the infected program and restore it to its original state. Remove the virus from all infected systems so that the virus cannot spread further.

If detection succeeds but either identification or removal is not possible, then the alternative is to discard the infected file and reload a clean backup version. Advances in virus and antivirus technology go hand in hand. Early viruses were relatively simple code fragments and could be identified and purged with relatively simple antivirus software packages. As the virus arms race has evolved, both viruses and, necessarily, antivirus software have grown more complex and sophisticated.

• First generation: simple scanners

• Second generation: heuristic scanners

• Third generation: activity traps

• Fourth generation: full-featured protection

A first-generation scanner requires a virus signature to identify a virus. The virus may contain "wildcards" but has essentially the same structure and bit pattern in all copies. Such signature-specific scanners are limited to the detection of known viruses. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length.

A second-generation scanner does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable virus infection. One class of such scanners looks for fragments of code that are often associated with viruses. For example, a scanner may look for the beginning of an encryption loop used in a polymorphic virus and discover the encryption key. Once the key is discovered, the scanner can decrypt the virus to identify it, then remove the infection and return the program to service. Another second-generation approach is integrity checking. A checksum can be appended to each program. If a virus infects the program without changing the checksum, then an integrity check will catch the change. To counter a virus that is sophisticated enough to change the checksum when it infects a program, an encrypted hash function can be used.

Third-generation programs are memory-resident programs that identify a virus by its actions rather than its structure in an infected program. Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of viruses. Rather, it is necessary only to identify the small set of actions that indicate an infection is being attempted and then to intervene.

Fourth-generation products are packages consisting of a variety of antivirus techniques used in conjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.

The arms race continues. With fourth-generation packages, a more comprehensive defence strategy is employed, broadening the scope of defence to more general-purpose computer security measures.

5.10 DISTRIBUTED DENIAL OF SERVICE (DDoS) ATTACKS

A denial of service (DoS) attack is an attempt to prevent legitimate users of a service from using that service. When this attack comes from a single host or network node, then it is simply referred to as a DoS attack. A more serious threat is posed by a DDoS attack. In a DDoS attack, an attacker is able to recruit a number of hosts throughout the Internet to simultaneously or in a coordinated fashion launch an attack upon the target.

DDoS attacks make computer systems inaccessible by flooding servers, networks, or even end user systems with useless traffic so that legitimate users can no longer gain access to those resources. In a typical DDoS attack, a large number of compromised hosts are amassed to send useless packets.

DDoS Attack Description

A DDoS attack attempts to consume the target's resources so that it cannot provide service. One way to classify DDoS attacks is in terms of the type of resource that is consumed. Broadly speaking, the resource consumed is either an internal host resource on the target system or data transmission capacity in the local network to which the target is attacked.

A simple example of an internal resource attack is the SYN flood attack. Figure 10.9a shows the steps involved:

a. The attacker takes control of multiple hosts over the Internet, instructing them to contact the target Web server.
b. The slave hosts begin sending TCP/IP SYN (synchronize/initialization) packets, with erroneous return IP address information, to the target.
c. Each SYN packet is a request to open a TCP connection. For each such packet, the Web server responds with a SYN/ACK (synchronize/acknowledge) packet, trying to establish a TCP connection with a TCP entity at a spurious IP address. The Web server maintains a data structure for each SYN request waiting for a response back and becomes bogged down as more traffic floods in. The result is that legitimate connections are denied while the victim machine is waiting to complete bogus "half-open" connections.

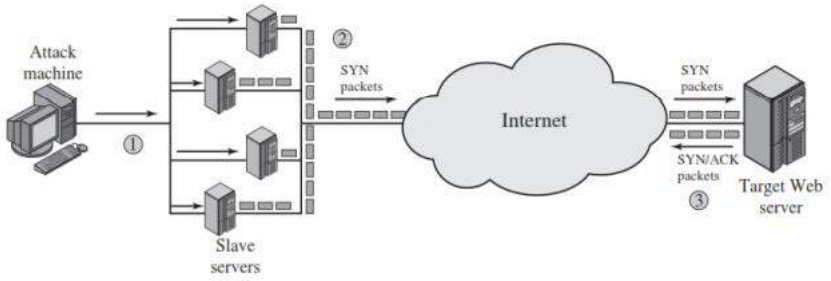


Fig. 5.1 Distributed SYN flood attack

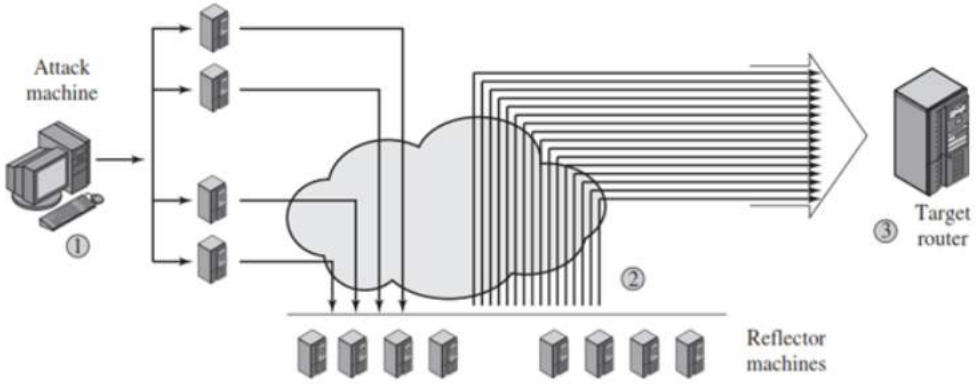


Fig. 5.2 Distributed ICMP attack

Figure illustrates an example of an attack that consumes data transmission resources. The following steps are involved:

i. The attacker takes control of multiple hosts over the Internet, instructing them to send ICMP ECHO packets with the target's spoofed IP address to a group of hosts that act as reflectors, as described subsequently.

ii. Nodes at the bounce site receive multiple spoofed requests and respond by sending echo reply packets to the target site.

iii. The target's router is flooded with packets from the bounce site, leaving no data transmission capacity for legitimate traffic

Another way to classify DDoS attacks is as either direct or reflector DDoS attacks. In a direct DDoS attack, the attacker is able to implant zombie software on a number of sites distributed throughout the Internet. Often, the DDoS attack involves two levels of zombie machines: master zombies and slave zombies. The hosts of both machines have been infected with malicious code. The attacker coordinates and triggers the master zombies, which in turn coordinate and trigger the slave zombies. The use of two levels of zombies makes it more difficult to trace the attack back to its source and provides for a more resilient network of attackers.

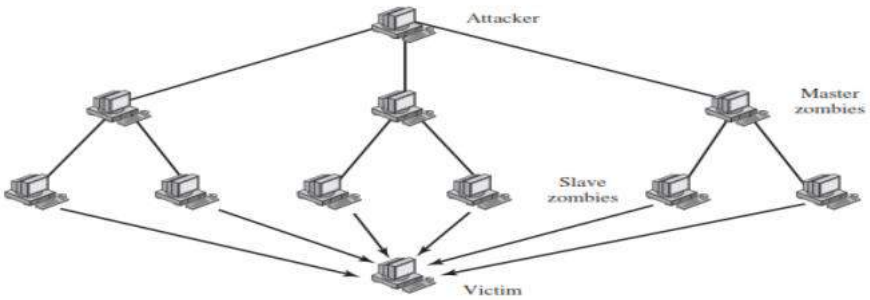


Fig. 5.3 Direct DDoS attack

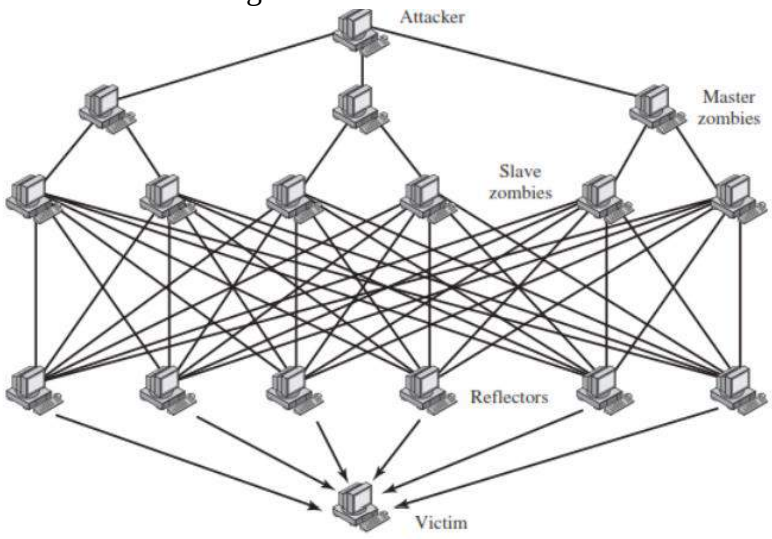


Fig. 5.4 Reflector DDoS attack

A reflector DDoS attack adds another layer of machines (Figure 10.10b). In this type of attack, the slave zombies construct packets requiring a response that contains the target's IP

address as the source IP address in the packet's IP header. These packets are sent to uninfected machines known as reflectors. The uninfected machines respond with packets directed at the target machine. A reflector DDoS attack can easily involve more machines and more traffic than a direct DDoS attack and hence be more damaging. Further, tracing back the attack or filtering out the attack packets is more difficult because the attack comes from widely dispersed uninfected machines.

5.11 DDoS Countermeasures

In general, there are three lines of defense against DDoS attacks.

- Attack prevention and pre-emption (before the attack): These mechanisms enable the victim to endure attack attempts without denying service to legitimate clients. Techniques include enforcing policies for resource consumption and providing backup resources available on demand. In addition, prevention mechanisms modify systems and protocols on the Internet to reduce the possibility of DDoS attacks.
- Attack detection and filtering (during the attack): These mechanisms attempt to detect the attack as it begins and respond immediately. This minimizes the impact of the attack on the target. Detection involves looking for suspicious patterns of behaviour. Response involves filtering out packets likely to be part of the attack.
- Attack source traceback and identification (during and after the attack): This is an attempt to identify the source of the attack as a first step in preventing future attacks. However, this method typically does not yield results fast enough, if at all, to mitigate an on-going attack.

5.12 INTRUDERS

One of the two most publicized threats to security is the intruder (the other is viruses), often referred to as a hacker or cracker. In an important early study of intrusion, it had been identified 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.

Some of the examples for intrusion are,

• 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

5.13 Intrusion detection systems

These have been developed to provide early warning of an intrusion so that defensive action can be taken to prevent or minimize damage.

Intrusion detection involves detecting unusual patterns of activity or patterns of activity that are known to correlate with intrusions.

Consumers commonly mistake an intrusion detection system (IDS)with a computer firewall. Although both applications have a similar goal to protect end-users from nefarious hackers and computer malware, an IDS differs from a firewall in that it can be either a device or software program created to monitor an individual computer, computing device, or network for either security policy violations or malicious activity. Once this type of behaviour is observed, the intrusion detection system makes a report to a centralized management component or station.

What is an Intrusion Detection System? Intrusion detection systems are designed to analyse network traffic for potentially malicious behaviour and to report possible "intrusions" to a centralized management node. Some IDSs are designed to take action to prevent these attempts from being successful; however, stopping malicious attacks is not a required component of an IDS. Many times, an organization will install an IDS to help document existing threats to company networks, to identify existing issues with violations of security policy, or to deter end-users from consistently violating company or organization security policies. Since IDSs were first introduced, they have become a critical component to most major organization's security infrastructures.

The concept of an intrusion detection system dates to 1984 when Fred Cohen determined that it was possible to detect network intrusions based on information available to network administrators if enough computing resources were devoted to the task. By taking a hard look at file access logs, user access logs, and system event logs, most unauthorized network intrusions could be detected.

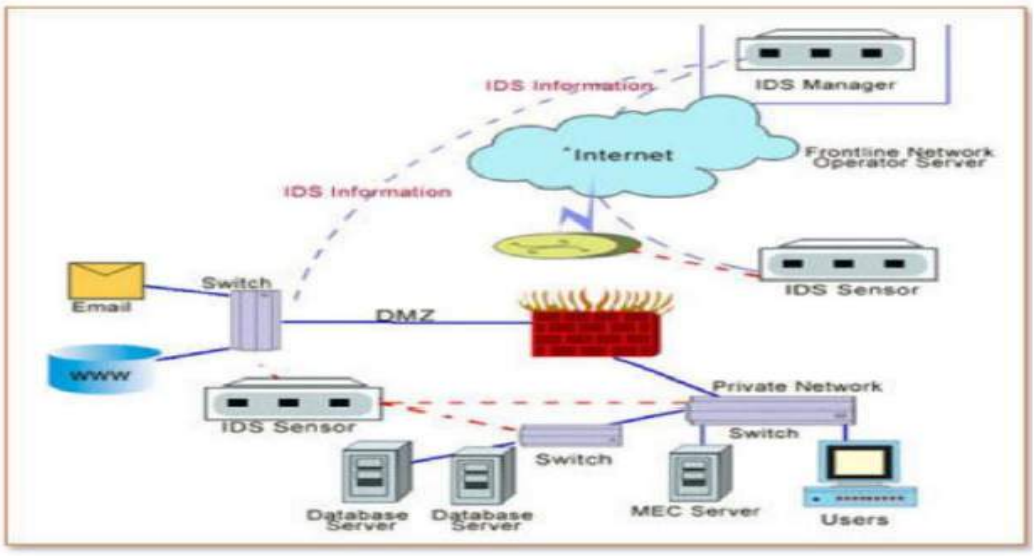


Fig. 5.5

Intrusion Detection Expert System

In 1986, Dorothy E. Denning assisted by Peter G. Neumann, published a new IDS model that continues to serve as the basis for intrusion detection systems in use today. Her model from the mid-1980's made use of statistical analysis for detecting network anomalies. The resulting implementation of this work was the Intrusion Detection Expert System (IDES) implemented at SRI International that ran on Sun work stations. This implementation made use of both rules set as well as a statistical anomaly detection system that looked at host systems, target systems, and end-users. Later, Lunt added an artificial neural network as a third component to the system which all made reports to a resolver application. The resulting work was deployed in the Next generation Intrusion Detection Expert System, or NIDES.

5.14 Types of Intrusion Detection Systems

There are three types of intrusion detection systems on the market today: network instruction detection systems (NIDSs), host-based instruction detection systems (HIDSs), and stack based intrusion detection systems (SIDS). Network Intrusion Detection System A network intrusion detection system analyses network traffic and hosts to locate potential intrusions. The NIDS system connects to a network hub, network tap, or network switch that is configured to allow monitoring of network traffic. When setting up a network intrusion detection system, the monitoring points are setup at high-traffic areas on the network to examine the network data packets for potentially malicious actions.

- Host-Based Intrusion Detection System:
  Host-based intrusion detection systems (HBIDs)are designed to have one network host agent that uses application logs, file-system modifications, and system call analysis to locate intrusions to the network. The sensors in a host-based intrusion detection system normally consist of software agent(s). A common example of a HIDS are OSSEC and Tripwire.
- Stack-Based Intrusion Detection System:
  Stack-based intrusion detection systems (SIDS) were developed as a succeeding technology to HBIDs. SIDS examine network packets as they travel through the network stack (TCP/IP). As a result, the SIDS technology does not incur the overhead of having to communicate with the network interface in promiscuous mode.

What are the Differences Between Statistical and Signature-Based Intrusion Detection Systems?

Statistics-based intrusion detection systems have been deployed for a number of years. This type of IDS will record normal network activity such as the types of protocols commonly used, devices connected to the network, ports used, and overall bandwidth. When network activity is detected that is out of the ordinary, the IDS will provide an alert to the network administrator or end-user regarding the event(s). A signature-based intrusion detection system compares network data packets with pre-determined network attack patterns or signatures. Unfortunately, there can be a significant delay in identifying new threat signatures to upload to the IDS. This makes signature-based IDSs vulnerable to emerging threats.

How Does an Intrusion Detection System Differ from a Firewall?

A common misconception amongst end-users is that firewalls and intrusion detection systems are the same thing. Although both technologies help preserve network and computer security,

they have distinct functions. Firewalls are designed to limit access from origins outside of the network to stop attacks from occurring. They are unable to identify malicious actions that being inside of the network. Intrusion detection systems are designed to identify attacks once they have gained access to the network and can evaluate potentially malicious actions which originate from within the network. As technologies have matured; however, a hybrid system referred to as an intrusion prevention system has been developed. The IPS is designed to stop malicious network connections and is also considered to be a firewall residing in the application layer of the OSI network model.

Honeypot:

In a fully deployed IDS, some administrators may choose to install a "honeypot," essentially a system component set up as bait or decoy for intruders. Honeypots can be used as early warning systems of an attack, decoys from critical systems, and data collection sources for attack analyses. Many IDS vendors maintain honeypots for research purposes, and to develop new intrusion signatures. Note that a honeypot should only be deployed when the organization has the resources to maintain it. A honeypot left unmanaged may become a significant liability because attackers may use a compromised honeypot to attack other systems.

Limitations of Intrusion Detection Systems:

Intrusion detection systems are not perfect. Depending on the design of the system, a number of false-positive results can be generated. These "false alarms" can originate from bad software, corrupt domain name server information, or local network traffic. As a result, a real network attack can be missed if the IDS is not properly configured for the defended network. Another vulnerability of IDSs that rely on signature files is updating the signature library to include the latest threats. When left undone, the network can be open to attack from the most current threats.

Free Intrusion Detection Systems:

There are several freely available intrusion detection / prevention systems available on the marketplace today. Some of the better-known projects include Snort, File System Saint, and AIDE.

- **Snort**:

One of the most downloaded and installed intrusion detection and prevention systems in the world today is Snort. Originally published in 1998 by CTO Martin Roesch, the application is designed to perform real-time packet logging and traffic analysis on IP-based networks. At the time of this writing, Snort has been downloaded more than four million times since initial release and has more than 400,000 registered users of the software. The application is based on a rule-based language that combines several additional IDS technologies to include protocol, anomaly-based, and signature detection methods.

- **File System Saint:**

File System Saint (FSS) is another open-source intrusion detection system written in the Perl programming language. The software project is designed to be lightweight, fast, and easy to use. FSS works on the basic premise of storing an image of the live file system of the network being protected and analyzes the system for any changes to the baseline report. The

application also stores data about file owner, permissions, file size, mtime, and ctime and reports changes to the computer owner via email report. To guard against tampering, FSS saves a cryptographic hash file to ensure legitimate data is being used while in operation.

- **AIDE:**

AIDE (Advanced Intrusion Detection Environment) is deployed as a free replacement for the commercially available Tripwire IDS. The software application is designed to check the integrity of the system's file and directories. To achieve this functionality, AIDE creates a database from the regular expression rules contained in the software's configuration files. After the database is created, it is used to validate the file integrity of the protected computer. Additional application features include support for the following message digest algorithms: sha1, rmd160, md5, crc32, sha256, sha512, tiger, and whirlpool. AIDE also supports gzip database compression if zlib support is installed on the protected computer.

5.15 IDS DEPLOYMENT (SNORT as example) Snort is logically divided into multiple components. These components work together to detect particular attacks and to generate output in a required format from the detection system. A Snort-based IDS consists of the following major components.

Table 5.2 Components of an IDS

Name	Description
Packet Decoder	Prepares packet for processing
Pre-processors or Input Plugins	Used to normalize protocol headers, detect anomalies, assembly and TCP stream re-assembly
Detection Engine	Applies rules to packets
Logging and Alerting System	Generates alert and log messages
Output Modules	Process alerts and logs and generate final output

Packet Decoder:

Packet decoder takes packets from different types of network interfaces and prepares packets to be pre-processed or to be sent to the detection engine. The interfaces may be Ethernet, SLIP, PPP, and so on.

Pre-processors:

Pre-processors are components or plug-ins that can be used with Snort to arrange or modify data packets before the detection engine does some operation to find out if the packet is being used by an intruder.

The Detection Engine:

The detection engine is the most important part of snort. Its responsibility is to detect if any intrusion activity exists in a packet. The detection engine employs snort rules for this purpose.

Logging and Alerting System:
Depending upon what the detection engine find inside a packet may be used to log the activity or generate an alert. Logs are kept in simple text files, tcp-dump-style files or some other form.

Output Modules:
Output modules or plug-ins can do different operations depending on how you want to save output generated by the logging and alerting system of Snort. Basically, these modules control the type of output generated by the logging and alerting system.

Figure below shows how these components are arranged. Any data packet coming from the Internet enters the packet decoder. On its way towards the output modules, it is either dropped, logged or an alert is generated.

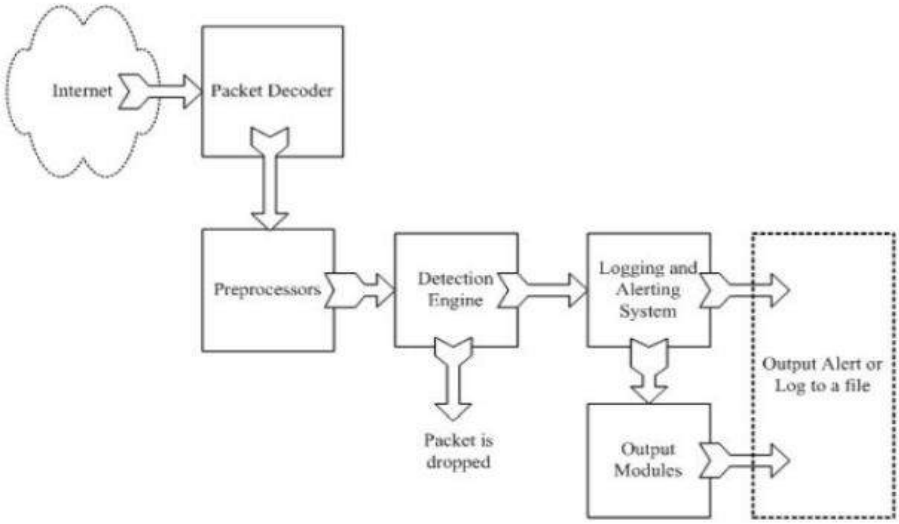


Fig. 5.6 Components of Snort

Review Questions:

1. Differentiate worms and viruses.
2. What are called as malicious software?
3. What do you mean by backdoor?
4. Define intruder.
5. Specify the antivirus approaches.
6. Define logic bomb.
7. Distinguish trojan horses and viruses.

8. List the types of intrusion detection systems.
9. What is DDoS attack?
10. Give the countermeasures of DDoS attack.
11. What are the various Computer Systems Security tools? Explain Malicious Software.
12. What is a Virus and Explain are the various types of Virus?
13. How is the computer system Security achieved using Virus & Antivirus?
14. What is denial of Service attack and How is it countered?
15. What is the difference between Network based & Host based IDS? Also Explain the steps for deploying the IDS?

REFERENCES:

1. Willliam Stallings," Network System Essentials "-4th Edition Copyright © 2011 Pearson education, Inc., publishing as Prentice Hall
2. Atul Khahate, "Cryptography and network security",3rd Edition, Copyright © 2013 TMH Publishing
3. Kuldeep Singh Kohar, "Network Security", revised reprint 2011.Vayu Education of India, New Delhi.