# Syllabus
# MCA Sem-IV, Paper - III
# Network Security

1. **Introduction:**
   Attacks, Services and Mechanisms, Security Attacks, Security Services, Integrity check, digital Signature, authentication, has algorithms.

2. **Secret Key Cryptography:**
   Block Encryption, DES rounds, S-Boxes
   IDEA: Overview, comparison with DES, Key expansion, IDEA rounds, Uses of Secret key Cryptography; ECB, CBC, OFB, CFB, Multiple encryptions DES.

3. **Hash Functions and Message Digests:**
   Length of hash, uses, algorithms (MD2, MD4, MD5, SHS) MD2: Algorithm (Padding, checksum, passes.) MD4 and 5: algorithm (padding, stages, digest computation.) SHS: Overview, padding, stages.

4. **Public key Cryptography:**
   Algorithms, examples, Modular arithmetic (addition, multiplication, inverse, and exponentiation) RSA: generating keys, encryption and decryption. Other Algorithms: PKCS, Diffie-Hellman, El-Gamal signatures, DSS, Zero-knowledge signatures.

5. **Authentication:**
   Password Based, Address Based, Cryptographic Authentication. Passwords in distributed systems, on-line vs off-line guessing, storing. Cryptographic Authentication: passwords as keys, protocols, KDC's Certification Revocation, Inter-domain, groups, delegation. Authentication of People:
   Verification techniques, passwords, length of passwords, password distribution, smart cards, biometrics.

6. **Security Policies and Security Handshake Pitfalls:**
   What is security policy, high and low level policy, user issues?
   Protocol problems, assumptions, Shared secret protocols, public key protocols, mutual authentication, reflection attacks, use of timestamps, nonce and sequence numbers, session keys, one-and two-way public key based authentication.

7. **Example System:**

Kerberos: purpose, authentication, serer and ticket granting server, keys and tickets, use of AS and TGS, replicated servers.

Kerberos V4: names, inter-realm authentication, Key version numbers.

Kerberos V5: names, realms, delegation, forwarding and proxies, ticket lifetimes, revoking tickets, multiple Realms.

8. **Network Security:**
   Electronic mail security, IP security, Network management security.

9. Security for electronic commerce: SSL, SET

10. **System Security:**
    Intruders and Viruses, Firewalls, Intrusion Detection

**Tutorials:**

1. Numerical problems on DE, IDEA, MD2, MD5, Diffie-Helman and El-Gamal Signatures.
2. Comparative study of network security tools.
3. Vulnerability tools: Nessus, Retina, Wireshark, Nmap.
4. Packet Sniffers: Tcpdump, Ettercap, Dsniff.

**Term Work:** Term work/Assignment: Each candidate will submit a journal in which at least 10 assignments based on the above syllabus and the internal test paper. Test will be graded for 10 marks and assignments will be graded for 15 marks.

**References:**

1. Atul Kahate, Cryptography and Network Security, McGraw Hill.
2. Kaufman, c., Perlman, R., and Speciner, M., Network Security, Private Communication in a public world, 2nd ed., Prentice Hall PTR., 2002.
3. Stallings, W.,.Cryptography and Network Security: Principles and Practice, 3rd ed., Prentice Hall PTR.,2003.
4. Stallings, W. Network security Essentials: Applications and standards, Prentice Hall, 2000.
5. Cryptography and Network Security; McGraw Hill; Behrouz A Forouzan.
6. Information Security Intelligence Cryptographic Principles and App. Calabrese Thomson.
7. Securing A Wireless Network Chris Hurley SPD.

❖❖❖❖

# 1

# INTRODUCTION

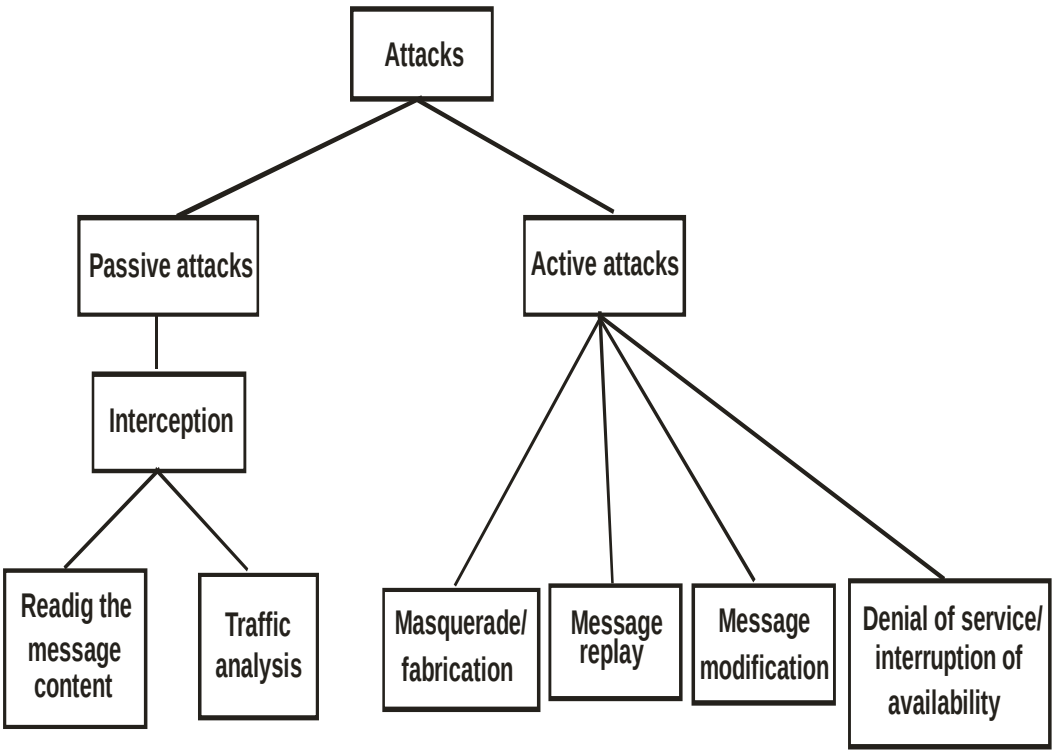
**In this unit, we are going to learn about the following topics:**

- 1.Attacks
  - ✓ 1.1. Active Attacks
  - ✓ 1.2. Passive Attacks
- 2.Security attacks
  - ✓ 2.1. Application Level
  - ✓ 2.2. Network Level
  - ✓ 2.3. Cookies
- 3.Services and mechanisms
- 4.Security services
  - ✓ 4.1. Confidentiality
  - ✓ 4.2. Authenticity
  - ✓ 4.3. Availability
  - ✓ 4.4. Auditability
  - ✓ 4.5. Access Control
  - ✓ 4.6. Integrity
  - ✓ 4.7. Non-repudiability
- 5. Integrity check
- 6. Digital signature
  - ✓ 6.1. Concept and Implementation
- 7. Authentication
- 8. Hash algorithms

## 1. ATTACKS:

In the cryptographic literature, there are two types of attacks namely Passive and Active. The first is a passive adversary, who can eavesdrop on all network communication, with the goal of learning as much confidential information as possible. The other is an active intruder, who can

- Modify messages at will,
- Introduce packets into the message stream, or
- Delete messages.




### 1.1. Active Attacks

- This type of attack requires the attacker to be able to transmit data to one or both of the parties, or block the data stream in one or both directions. It is also possible that the attacker is located between the communicating parties. The attacker even can stop all or parts of the data sent by the communicating parties. This attacker can try to take the place of the client (or server) when the authentication procedure has been performed. As the server will not detect the origin of the data without the help of integrity check mechanism, a clever programmer can, with not too much of effort, implement a system like a gateway (bridge) between two subnets. (On the Internet there are thousands of these computers.)

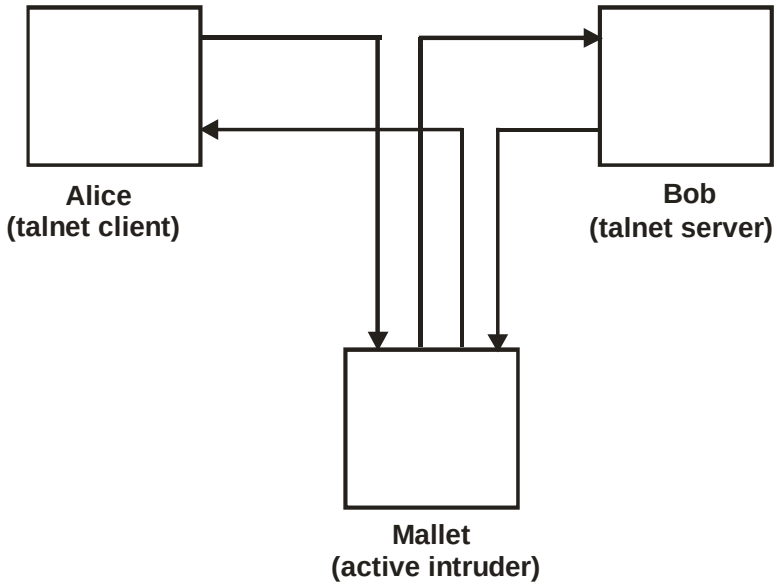



**Figure 1.0: Active attack**

The following are examples of different attacks this person could impose.

- **Inserting his own data into the data stream.**
- **Playback of data from another connection.**
- **Playback of data that had previously been sent in the same and opposite direction on the same connection.**
- **Deletion of data.**
- ***Man-in-the-middle* attack** -In this attack, the intruder sits in the middle of the communication link, intercepting messages and substituting them with his own messages. In this way, he tries to fool the parties to believe they are talking to each other directly, while they really are talking to the attacker him-selves.

There are three basic levels of issues, which will lead to the further loopholes. They are:

- **Interruption,**
- **Modification and**
- **Fabrication**

### 1.1.1 Interruption:

This attack can also be called as **masquerade attack**. This attack is possible by the user who will be pretending like a legitimate user and attack the system. For example any employee is sending the message to the working staff members by taking the name of his manager. He can send the message for conducting

meetings in a particular time with his employee, which is actually a forged message. It relates with the reputation of the users.

### 1.1.2. Modification:

Modification can be further divided into two categories.

1. **Replay attacks**
2. **Alterations**

Both the above-mentioned attacks are dealing with the **confidentiality** of the message. If the changes are made and the reflection is not projected in the original copy then it relates to the integrity problem of the same message.

**Replay attacks** can take place by sending the same message twice. For example when

User A sends a message stating that transfer Rs.10, 000 from Account A to Account B. This message is watched by another user C and the same message is sent once again by user C on behalf of A. In that scenario, Rs.20, 000 will be debited from account A instead of Rs.10, 000, without his notification. This transaction will create an integrity problem in such a way that the account detail with A shows Rs10, 000 as balance and bank transaction says the account balance of A is zero.

### 1.1.3. Fabrication:

Denial of Service attack is belonging to this category.

**Denial – of – Service Attacks:**

A **denial-of-service attacks** or **DoS attack** is an attempt to make a computer resource unavailable to its intended users. The basic purpose of a Denial of Service (DoS) attack is simply to flood/overhaul a network so as to deny the authentic users' services of the network. A DoS attack can be launched in many ways. The end result is the flooding of a network, or change in the configurations of routers on the network. In general terms, DoS attacks are implemented by either forcing the targeted computer(s) to reset, or consume its resources so that it can no longer provide its intended service or obstructing the communication media between the intended users and the victim so that they can no longer communicate adequately.

It is not easy to detect a DoS attack because there is nothing apparent to suggest that a user is launching a DOS attack, and is actually not a legal user of the system. This is because in a DoS attack, it is up to the server to detect that certain packets are from

an attacker, and not from a legitimate user to take an appropriate action. This is not an easy task. Failing this, the server would fall short of resources (memory, network connections, etc.) and come to a grinding halt after a while.

**1.1.3.1 Various other forms of DoS attack is given below.**

- **Buffer Overflow Attacks**
- **SYN Attack**
- **Teardrop Attack**
- **Smurf Attack**

**1.1.3.1.1 Buffer Overflow Attacks:**

It is the most common kind of DoS attack. It is simply to send more traffic to a network address. The attacker may be aware that the target system has a weakness that can be exploited or the attacker may simply try the attack in case it might work. A few of the better-known attacks based on the buffer characteristics of a program or system include:

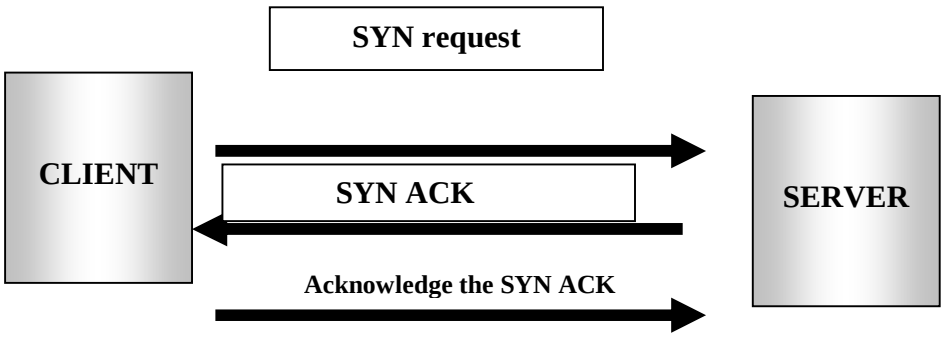
- Sending e-mail messages that have attachments with 256-character file names to Netscape and Microsoft mail programs.
- Sending oversized Internet Control Message Protocol (ICMP) Packets(this is also known as the Packet Internet or Inter-Network Groper(ping) of death)
- Sending to a user of the Pine e-mail program a message with a "From" address larger than 256 characters.

**1.1.3.1.2. SYN Attack**

When a session is initiated between the Transport Control Program (TCP) client and server in a network, a very small buffer space exists to handle the usually rapid "hand-shaking" exchange of messages that sets up the session. The session-establishing packets include a SYN field that identifies the sequence in the message exchange. An attacker can send a number of connection requests very rapidly and then fail to respond to the reply. This leaves the first packet in the buffer so that other, legitimate connection requests can't be accommodated. Although the packet in the buffer is dropped after a certain period of time without a reply, the effect of many of these bogus connection requests is to make it difficult for legitimate requests for a session to get established. In general, this problem depends on the operating system providing correct settings or allowing the network administrator to tune the size of the buffer and the timeout period. The client sends a SYN

(synchronization) request to the server which indicates that the client is requesting for a TCP connection with the server. Sample SYN Attack scenario is explained below.

1. The server responds back to the client with an acknowledgement, which is technically called as SYN ACK.
2. The client is then expected to acknowledge the server's SYN ACK. This is shown in the Figure below:




- The completion of these steps indicates that a TCP connection has been established between the client and server and is ready for exchange of data.
- Here is the junction point where an attacker would create a DoS attack keeping in mind the three basic steps required to establish a TCP connection.
- The attacker as a client would purposely go on performing step 1 wherein it will go on sending requests but will not acknowledge later in step 3 for any of those packets, which the server has sent as acknowledgement. This would result degrade the performance of the server and would come to a halt.

#### 1.1.3.1.3. Teardrop Attack

This type of denial of service attack exploits the way that the Internet Protocol (IP) requires a packet that is too large for the next router to handle be divided into fragments. The fragment packet identifies an offset to the beginning of the first packet that enables the entire packet to be reassembled by the receiving system. In the teardrop attack, the attacker's IP puts a confusing offset value in the second or later fragment. If the receiving operating system does not have a plan for this situation, it can cause the system to crash.

#### 1.1.3.1.4. Smurf Attack

In this attack, the executor sends an IP ping (or "echo my message back to me") request to a receiving site the ping packet

specifies that it be broadcast to a number of hosts within the receiving site's local network. The packet also indicates that the request is from another site, the target site that is to receive the denial of service. (Sending a packet with someone else's return address in it is called spoofing the return address.) The result will be lots of ping replies flooding back to the innocent, spoofed host. If the flood is great enough, the spoofed host will no longer be able to receive or distinguish real traffic.

#### 1.1.3.1.5Physical Infrastructure Attacks

Here, someone may simply snip a fiber optic cable. This kind of attack is usually mitigated by the fact that traffic can sometimes quickly be rerouted.

**Distributed Denial of Service Attack:**

- Distributed denial of service attack (DDoS) occurs when multiple compromised systems flood the bandwidth or resources of a targeted system, usually one or more web servers. These systems are compromised by attackers using a variety of methods.
- The major advantages to an attacker of using a distributed denial-of-service attack are that multiple machines can generate more attack traffic than one machine, multiple attack machines are harder to turn off than one attack machine, and that the behavior of each attack machine can be stealthier, making it harder to track down and shut down. These attacker advantages cause challenges for defense mechanisms.
- It is important to note the difference between a DDoS and DoS attack. If an attacker mounts an attack from a single host it would be classified as a DoS attack. On the other hand, if an attacker uses a thousand zombie systems to simultaneously launch smurf attacks against a remote host, this would be classified as a DDoS attack

### 1.2.Passive attacks

A passive attack on a cryptosystem is one in which the cryptanalyst cannot interact with any of the parties involved, attempting to break the system solely based upon observed data (i.e. the cipher text). This can also include known plaintext attacks where both the plaintext and its corresponding cipher text are known**.**

The passive attacks can take place in the following ways:

- **Eavesdropping:** The unauthorized capture of transmitted data either by some form of line tapping or

from the compromising emanations broadcast by the electrical signals in the line is known as eavesdropping. Radio, optical and microwave signals can be similarly intercepted covertly.

- **Traffic Analysis**: An analysis of the traffic down the line can, in many circumstances, reveal much to an outsider. The number, size, frequency and times of messages sent, their sources and their destination can be leaked from these types of analysis.
- A passive attack is an attack where an unauthorized attacker monitors or listens in on the communication between two parties. The figure below illustrates a passive attack where Eve monitors the communication between Alice and Bob.

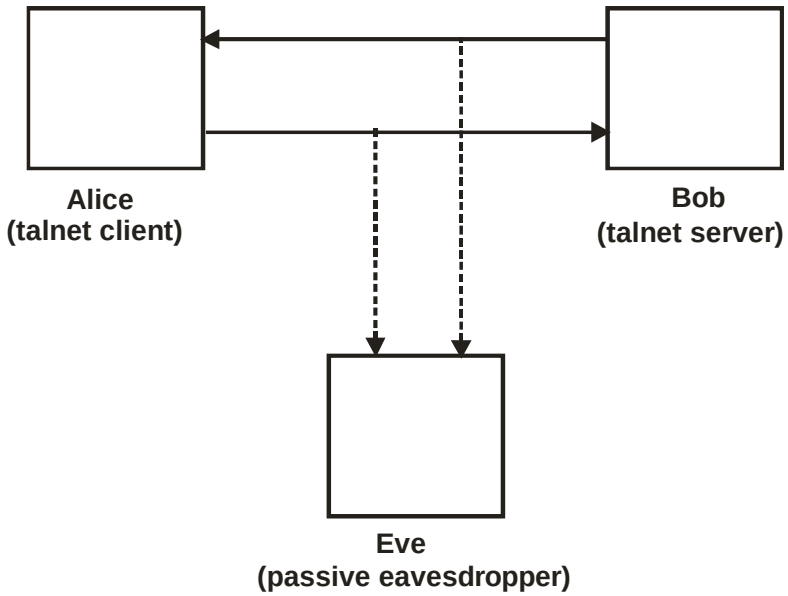



**Figure 1.1: Passive attack**

**2. There are two levels of practical side attacks.**

**(a) Application b) Network Level Attacks**

**Cookies are also come under practical side attacks.**

### 2.1 Application Level Attacks:

These attacks happen at an application level in the sends that the attacker attempts to access, modify or prevent access to information of a particular application, or the application itself. Example of this are trying to obtain someone's credit card information on the Internet, or' changing the contents of a message to change the amount in a transaction, etc.

## 2.2 Network level attacks:

These attacks generally aim at reducing the capabilities of a network by a number of possible means, these attacks generally make an attempt to either slow down, or completely bring to halt, a computer network. Note that this automatically can lead to application level attacks, because once someone is able to gain access to a network, he/she is able to access/modify at least some sensitive information.

## 2.3. Cookies:

The Internet uses HTIP protocol, which is stateless. Suppose that the client sends an HTIP request for a Web page to the server. The web server locates that page on its disk, sends it back to the client, and completely forgets about this interaction. If the client wants to continue this interaction, it must identify itself to the server in the next HTIP request. Otherwise, the server would not know that this same client and sent an HTIP request earlier. Since a typical application is likely to involve a number of interactions between the client and the server, there must be some mechanism for the client to identify itself to the server each time it sends a HTIP request to the server. For this, cookies are used. They are a popular mechanism of maintaining the state information i.e. identifying a client to a server. A cookie is just one or more pieces of information stored as text strings in a text file on the disk of the client computer i.e. Web browser.

**These attacks take two main forms:**

**(a) Packet Sniffing (also called as snooping)**

**(b) Packet Spoofing.**

Since the protocol used in this communication is called as Internet Protocol (IP), other names for these two attacks are: (a) IP sniffing and (b) IP spoofing. The meaning remains the same.

**Distinguish between packet sniffing/IP Sniffing and packet spoofing/IP Sppofing.**

**(a) Packet Sniffing:**

Packet sniffing is a passive attack on an ongoing conversation. An attacker need not hijack a conversation, but instead, can simply observe i.e. sniff packets as they pass by. Clearly, to prevent an attacker from sniffing packets, the information that is passing needs to be protected in some ways. This can be done at two levels:

i. The data that is traveling can be encoded in some ways.

ii. The transmission link itself can be encoded.

**(b) Packet Spoofing:**

In this technique, an attacker sends packets with an incorrect source address. When this happens, the receiver i.e. the party who receives the packets containing a false source address would inadvertently send replies back to the forged address (called as spoofed address) and not to the attacker.

This can lead to three possible cases:

i. The attacker can intercept the reply- If the attacker is between the destination and the forged source, the attacker can see the reply and use that information for hijacking attacks.

ii. The attacker need not see the reply-If the attacker's intention was a Denial of Service (DOS) attack, the attacker need not bother about the reply.

iii. The attacker does not want the reply- The attacker could simply be angry with the host. So it may put that host's address as the forged source address and send the packet to the destination. The attacker does not want a reply from the destination, as it wants the host with the forged address to receive it and get confused.

**Explain the security services and mechanisms:**

**3. Services and Mechanisms:**

***4. Communication Security Services***

- Confidentiality
  - Data Confidentiality
  - Traffic Confidentiality
- Data Integrity
- Authentication
  - Data Origin Authentication
  - Peer Authentication
- Access Control
- Non-Repudiation
  - Non-Repudiation of Origin
  - Non-Repudiation of Reception
- Audit
- Availability – an after-thought but increasingly important

### 4.1 Confidentiality:

It is the protection of information from disclosure to unauthorized entities such as organizations, people, machines and processes. The information includes data contents, size, existence, communication characteristics, etc.

#### 4.1.1. Protection Mechanisms

- Data Encryption
  - Symmetric (Secret-Key)
  - Asymmetric (Public-Key)

### 4.2 Integrity:

It is the protection of data against creation, alteration, deletion, and duplication, re-ordering by unauthorized entities such as organizations, people, machines and processes.Integrity violation is always caused by active attacks.

#### 4.2.1 Protection Mechanisms

- Message Digests (Hashing)
- Sequence Numbers
- Nonce ID (Random Number)
- Time Stamps

### 4.3. Authentication:

Communicating entities are provided with assurance & information of relevant identities of communicating partners such as people, machines and processes.Personnel Authentication requires special attention.

#### 4.3.1. Protection Mechanisms

- Password
  - Manual
  - One-Time Password
- Key Sharing
  - Manual
  - Symmetric Key (Tickets)
  - Asymmetric Key (Certificates)
- Challenge – Response
  - Nonce Based
  - Zero Knowledge Proof

### 4.4. Access Control:

This is the protection of information resources or services from access or use by unauthorized entities (organizations, people, machines, processes).

- ❖ Privileges – rights to access or use resources or services
- ❖ Principles – entities own access control privileges
- ❖ Subjects – entities exercise access control privileges
- ❖ Objects / Targets – resources or services accessed/used by subjects
- ❖ Delegation – transfer of access control privileges among principals
- ❖ Authorization – transfer of access control privileges from principals to subjects

### 4.4.1. Protection Mechanisms

- ❖ Access Control Lists (ACLs)
  - Object Based Specification-Ex.: UNIX File System
- ❖ Capabilities
  - ❖ Subject Based Specification
  - ❖ Issue Tickets/Certificates

## 4.5 Non-Repudiation:

This service is the protection against denial of participation by communicating entities in all or part of a communication

### 4.5.1. Protection Mechanisms

- o Time Stamp
- o Digital Signature

## 4.6. Audit:

Recording & analyses of participation, roles and actions in information communication by relevant entities is known as audit.

### 4.6.1. Protection Mechanisms

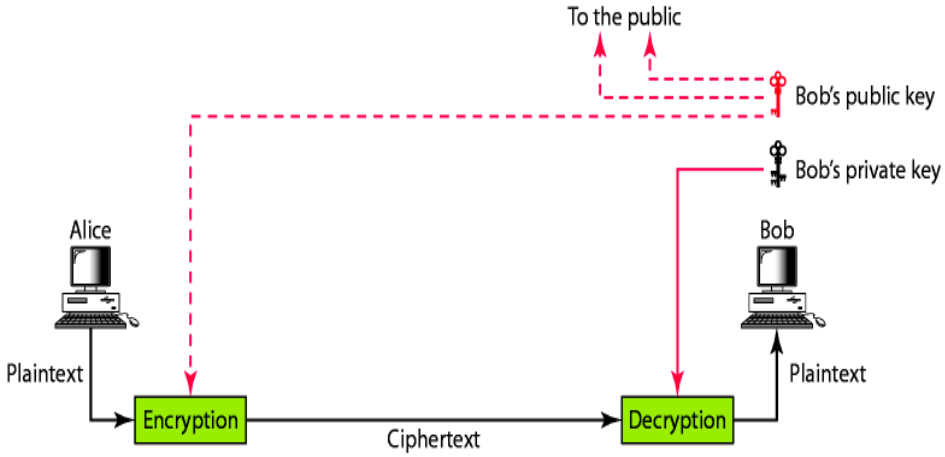
- ❖ “Syslog”
- ❖ Intrusion Monitors / Sensors
  - ❖ Common Intrusion Detection Framework (CIDF)
  - ❖ Common Information Model (CIM)

**3. Well known Security Mechanisms are listed below:**

- Encipherment – with Secret / Public Key Cryptography
- Data Integrity – with One-Way Hash Function
- Authentication – with Public-Key Challenge/Response
- Access Control
- Digital Signature – with Public-Key Cryptography
- Traffic Padding

### 3.1. Secret/Public key cryptography:

In public-key cryptography, there are two keys: a private key and a public key. The receiver keeps the private key. The public key is announced to the public. Imagine Alice wants to send a message to Bob. Alice uses the public key to encrypt the message. When Bob receives the message, the private key is used to decrypt the message. In public-key encryption/decryption, the public key that is used for encryption is different from the private key that is used for decryption. The public key is available to the public; the private key is available only to an individual.



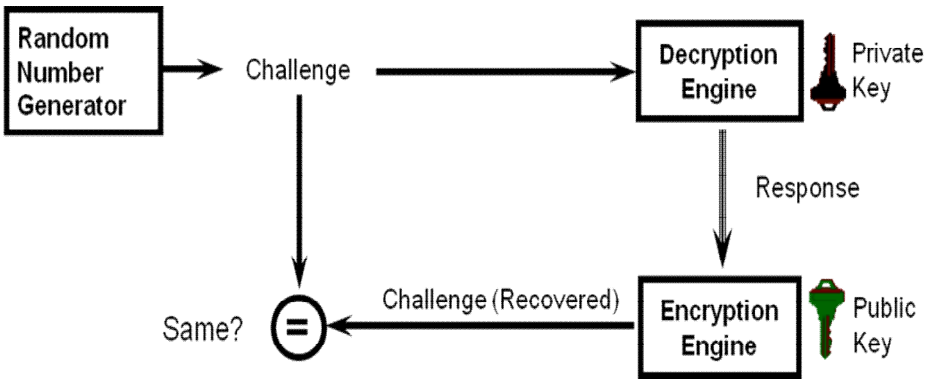

### 3.1.1. One-Way Hash Function:

One way hash - A fixed length output string is generated from a variable length input. The reverse of the function is almost impossible to perform. A one way hash is also called a message digest, cryptographic checksum, message integrity check (MIC), and manipulation detection code (MDC).

### 3.2. Public Key Challenge-Response Authentication:

- Challenger sends **a *challenge*** of random number to Responder

- Responder creates a ***response*** by digitally signing the challenge with its private key and returns the response to the Challenger
- Challenger processes the response with public key of legitimate Responder and compare it with original challenge




**Single security services may need to be implemented by multiple and different security mechanisms.**

**Compare the security services and security mechanisms:**

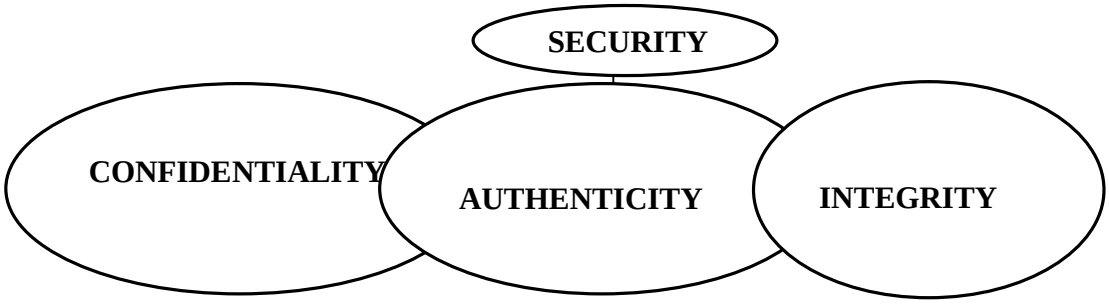
| Security Services | Security Mechanisms |
|---|---|
| Data Confidentiality | **Mechanism I:Encipherment**<br>Symmetric (Secret-Key) Cipher<br>Asymmetric (Public-Key) Cipher |
| Traffic flow confidentiality | **Mechanism I:Encipherment**<br>Traffic padding, routing control |
| Data Integrity | **Mechanism II: One-Way Hash Function**<br>Message Digest / Digital Hash |
| Authentication | **Mechanism III: Challenge + Response**<br>Nonce (Random Number) Based Zero-Knowledge Proof |
| Nonrepudiation | **Mechanism II: One-Way Hash Function**<br>Digital signature, data integrity, notarization |

| | |
|---|---|
| Access control | Access control |
| Availability | Data integrity and authentication exchange |

**3.3 Security services /Principles of security**:

Generally the security issues can be classified into the following categories.

1. Confidentiality
2. Authenticity
3. Availability
4. Auditability
5. Access Control
6. Integrity
7. Non-repudiability




These three are the basic levels of issues, which will lead to the further loopholes.

**3.3.1 Confidentiality:**

Basically the threats can be in the area of network or in the area of application. Mainly the insiders who are having full access with the computer system create the application levels of threats. This can be easily detected can be avoided by using suitable mechanisms. Even though the application level threats are easy to detect, this is also creating high level of problems to the system. The network levels of attacks are dangerous because of the major business transactions that are happening through the Internet. This is maintaining the actual secrecy of the message. The message that is traveling through the network should not be opened by any of the third parties who are not related with the transaction.

Nowadays majority of the bank transactions are happening through the network. So that confidentiality issue is creating lot of problems related with network, software and data.

The confidentiality related issues could be belonging to any one of the following categories:

1. IP Spoofing
2. Packet sniffing
3. Alteration of message
4. Modification of message
5. Man-in-middle
6. Brute force attack
7. Password cracking

### 3.3.2. Authenticity:

This can be defined as an identity for a user to assure that the message is coming from the right person. This is also an important issue along with confidentiality which may lead to the further security threats. This can be assured by any of the following factors:

1. **Something you have (like tokens, credit card, passport etc).**
2. **Something you know (like PIN numbers, account number etc).**
3. **Something you are (like fingerprints, signatures etc).**

Generally with the computer system passwords are the very simple authentication mechanism, which help the system to authenticate a particular person. The people can use one-time passwords and key technology to assure authenticity during message transaction.

Various issues related to authenticity includes

1. Stealing password
2. Fake login screen
3. Information leakage Etc.

### 3.3.3 Availability:

This can be defined as keeping the right information or resources available to the right person at the right time. This can happen either with the data or with the hardware resources. This

will stop the person from accessing various resources by flooding the network.

There is a complexity with the availability of the resources and data. Because it can be identified as a issue only when the following conditions are existing in the system.

1. **The resources are completely available up to the users expectation.**
2. **The content is present in a usable format.**
3. **The access rights are used in a proper way.**

The actual problem related with availability can be identified only when the above-mentioned things are assured before the problem identification. This is a very serious issue, which will totally stop the process and may lead the user to an idle condition without allowing him to proceed with the further process. The main problem related with the availability is **DOS attack & DDOS attacks.** Flooding the network path by sending continuous packets, this will create a heavy traffic in the network, does this. This stops the right people accessing right information at the right time.

## 3.4. Access Control:

This is an issue, which is dealing with the hardware resources, software and data. This is helping the operating system to allow access to a particular resource or data only to an authorized person. In this way it is interrelated with the authenticity and availability. Because the authenticated users will be allotted with certain kinds of rights like **Read, Write, Read/Write, Owner** etc. These rights will be maintained by the operating system in a tabular format or in a linked list format. First the users' authenticity has to be verified and then the authentic users' rights will be verified against the table.

| USERS/FILES | FILE1 | FILE2 | FILE3 |
|---|---|---|---|
| USER1 | RW | - | W |
| USER2 | O | RW | W |
| USER3 | - | OR | OW |

The attacks related to authenticity and availability can also create the problem related with access control. Attacks related to access control are as follows

1. Intrusion
2. DDOS
3. Interference
4. Inference Etc.

**The issues related with authenticity can be resolved by using hash algorithms.**

## 3.5. Non-repudiability:

This is another issue, which is related with authenticity and integrity. Repudiability means refusing. This is an issue, which is actually created by the sender who is participating in the transaction. After sending a message a sender can refuse that he was not sending that message. This is done intentionally to create problems at the receiver's side, which creates confusion to the receiver. This can be done from either side. It may also be from the receiver side. The receiver can deny after receiving the message that he doesn't receive any message. The non-repudiability related issues can happen in any of the following three ways.

1. **Proof of origin**
2. **Proof of receipt**
3. **Proof of content**

This can be assured by using digital signatures along with the hash algorithms. If the proper authenticity and integrity is achieved then the problems related with non-repudiability can be minimized. All above-mentioned issues are the basic issues of network security. Apart from these various other threats like natural disasters, attacks, software modifications are also creating problems with networks. But majority of the attacks are coming under the basic issues of the network. Wherever the problems are available accordingly solutions are also present. It is the responsibility of the user to categorize the problem and identify the suitable solution.

Generally the solutions can be categorized as follows.

a) Security Issues
b) Security Objectives
c) Security Techniques

| SECURITY ISSUE | SECURITY OBJECTIVE | SECURITY TECHNIQUE |
|---|---|---|
| **Confidentiality** | Privacy of message | Encryption |
| **Authentication** | Origin Verification | Digital Signatures<br>Challenge-response<br>Passwords<br>Biometric devices |

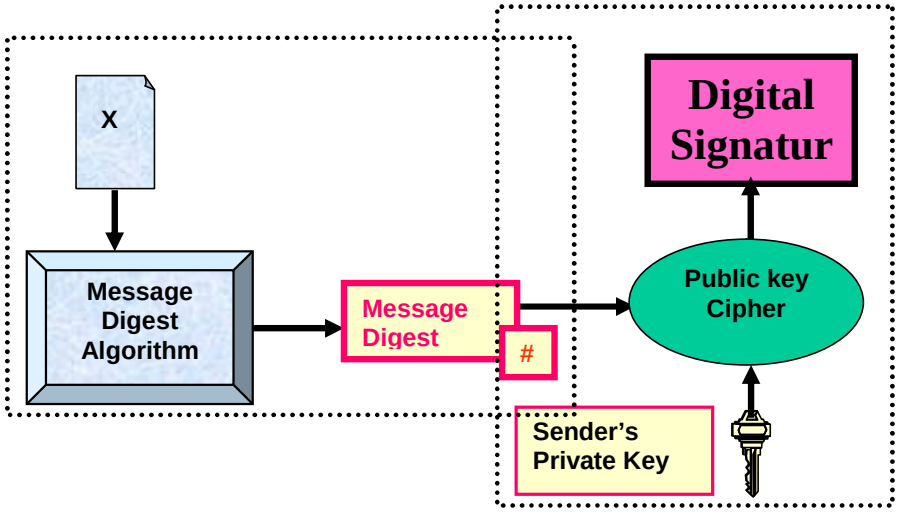
| | | |
|---|---|---|
| **Non-repudiation** | Proof of origin, receipt and contents | Bi-directional hashing<br>Digital signatures<br>Transaction Certificates<br>Time stamps<br>Confirmation services |
| **Access controls** | Limiting entry to authorized users | Firewalls<br>Passwords<br>Biometric devices |

**Explain the concept of Digital Signature:**

**7. Digital Signatures:**

Consider the situation in which if X is sender & Y receiver, then X encrypts the message with Y's public key and on receiving, Y decrypts with his own private key. This method only ensures secure communication between the two. Now consider another situation. If X is sender and Y is receiver, X encrypts the message using his own private key! On receiving, Y decrypts it using X's public key. The purpose behind this move is 'authentication'. It is clear that, only X knows his private key. So, when Y receives this message (encrypted with X's private key), it is an indication or proof that it has originated only from X and none else! Remember that in earlier scheme, the purpose was only 'confidentiality' and the origin of message was not the concern. Now, one may say that if someone else wants to intercept this communication it should be easy. That is anyone can decrypt the message who knows X's public key. This is true, but then it will not be possible for anyone to again encrypt this message as only X knows his private key. Thus receiver here will not be fooled that message came from X This scheme confirms the origin of the message. So, in this case X cannot deny that he has sent the message to Y, because it was encrypted with X's private key, known only to X.

The above discussion forms the basis for the concept called **"Digital Signature''** In case of our normal operations, we make use of our (handwritten) signatures. These are used to confirm the 'origin' or the 'authentication' of the individual. In the Internet world, it would be difficult to use any such method in practice. Hence the concept of 'Digital signatures' was evolved. This technique is vitally important in the E-commerce concept used in the Internet. It proves as a valid mechanism for 'authenticity' of individual. Most of the financial transactions done over Internet make use of this method.




### 7.1. Techniques of Digital signatures:

Actual working of Digital signatures involves the use of a concept called 'Message digest' or 'hash'. Message digest is something like the summary of original message. (works similar to the CRC checksum concept) This is basically used to verify the 'integrity' of data i.e. to ensure that the message has not been modified after it was sent by sender and before it reaches the receiver. The Digital Signature Standard (DSS) was developed by NIST first in 1991. It suggests using the SHA-1 algorithm for calculating the message digest. This digest is further used for performing Digital signatures, by using the algorithm called Digital Signature Algorithm (DSA). In DSA, message digest is encrypted with the sender's private key to form the Digital Signature (DS). This signature is transmitted further along with the original message. It is also possible to use the earlier RSA algorithm for performing digital signatures. RSA is prominently used over DSA as DSA turns out to be more complicated.

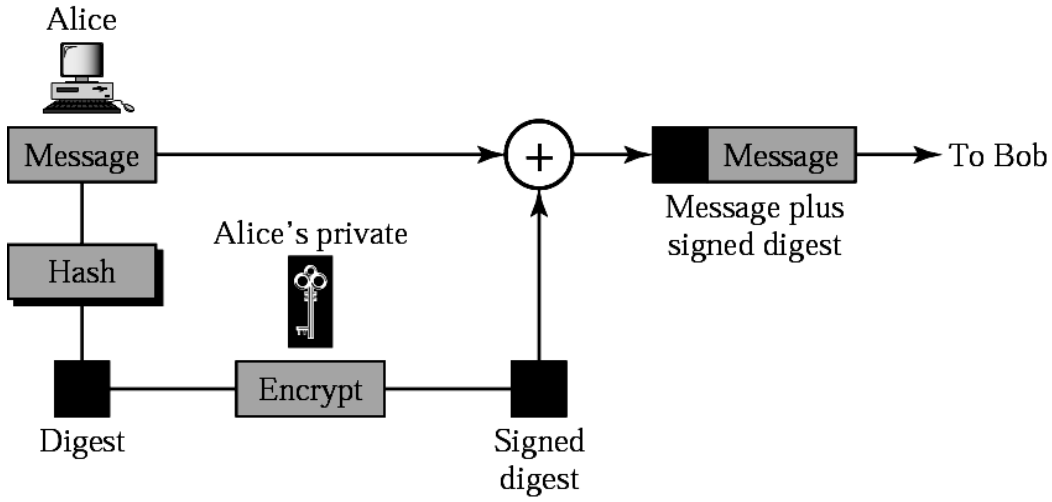
## 5. Integrity Check:

Integrity check is ensuring that the receiver of the message should be able to tell the message was not modified. To achieve this key exchange and digital signature are used. 5.1. **Implementation of Integrity Check through Digital Signature: (both sender and receiver side)**

**Sender's Side:**

1. If X is the sender, the SHA-1 algorithm is used to first calculate the message digest (MD 1) of original message.
2. This MD1 is further encrypted using RSA with X's private key. This output is called the Digital Signature (DS) of X.

3. Further, the original message (M) along with the Digital signature (DS) is sent to receiver.
4. After the digest has been created, it is encrypted (signed) using the sender's private key. The encrypted digest is attached to the original message and sent to the receiver.



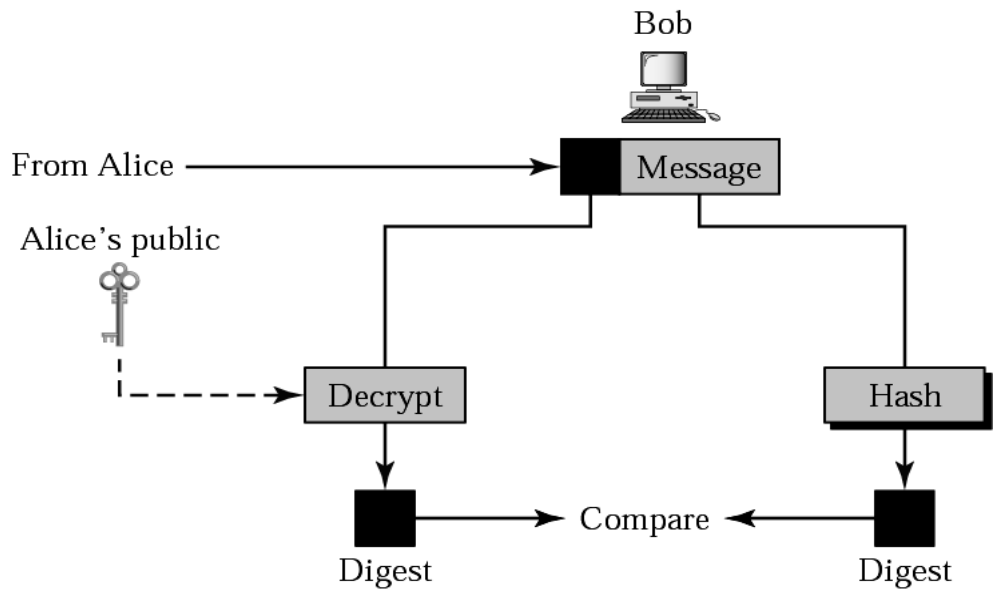

**Receiver's Side:**

1. Y thus receives the original message (M) and X's digital signature. Y uses the same message digest algorithm used by X to calculate the message digest (MD2) of received message (M).

2. Also, Y uses X's public key to decrypt the digital signature. The outcome of this decryption is nothing but original message digest (MD1) calculated by X.

3. Y then compares this digest MD1 with the digest MD2 he has just calculated in step 4. If both of them are matching, i.e. MDl = MD2, Y can accept the original message (M) as correctly authenticated and assured to have originated from X. whereas, if they are different, the message shall be rejected.

The implementation is depicted below.




This method turns out to be foolproof. Even if an attacker intercepts anywhere in between, it is not likely for him to again sign the modified/read message, as only X in this case will know the private key! Hence, even if intercepted, this method remains very much secure and reliable!

**5.2. Properties of Digital Signature:**

Digital signature does not provide privacy. If there is a need for privacy, another layer of encryption/decryption must be applied.

**Digital signatures can provide**
1. Integrity, Authentication, and Nonrepudiation.

1. **Integrity** The integrity of a message is preserved because if Eve intercepted the message and partially or totally changed it, the decrypted message would be unreadable.

2. **Authentication** We can use the following reasoning to show how a message can be authenticated. If Eve sends a message while pretending that it is coming from Alice, she must use her own private key for encryption. The message is then decrypted with the public key of Alice and will therefore be non-readable. Encryption with Eve's private key and decryption with Alice's public key result in garbage.

3. **Nonrepudiation** Digital signature also provides for non-repudiation. Bob saves the message received from Alice. If Alice

later denies sending the message, Bob can show that encrypting and decrypting the saved message with Alice's private and public key can create a duplicate of the saved message. Since only Alice knows her private key, she cannot deny sending the message.

**6. Authentication**

The receiver of the message should be able to be sure of the origin of the message. It requires a digital signature (One way hash, public key algorithm, and symmetric algorithm) or a public key algorithm. The following functions are associated with authentication service.

- **Salting** is the act of combining a user password with a random value as a countermeasure to a dictionary password attack. The combined values are encrypted and stored on the system along with the salt value. A random number string is combined with the user password before a one way function is applied to it. The salt value and the one way function result are stored on the host that authenticated the user. The password is not stored on the host.

- **SKEY** is a one way function authentication mechanism. A limited set of numbers are generated by using the function on a random number, then use the function on each result for a given number of times. The generated numbers are used only once for login.

**8. Hash Function:**

A **hash function** is any well-defined procedure or mathematical function that converts a large, possibly variable-sized amount of data into a small datum, usually a single integer that may serve as an index to an array. The values returned by a hash function are called **hash values**, **hash codes**, **hash sums**, or simply **hashes.** Hash functions are mostly used to speed up table lookup or data comparison tasks — such as finding items in a database, detecting duplicated or similar records in a large file, finding similar stretches in DNA sequences, and so on.A hash function may map two or more keys to the same hash value. In many applications, it is desirable to minimize the occurrence of such collisions, which means that the hash function must map the keys to the hash values as evenly as possible. Depending on the application, other properties may be required as well. Although the idea was conceived in the 1950s the design of good hash functions is still a topic of active research.

More about hash functions and hash algorithms will be dealt in Unit-3.

**Questions for practice:**

1) What is the classification of attacks? Compare and contrast both the types.
2) Explain about active attacks in detail.
3) Explain about passive attacks in detail.
4) What are the variations of Dos attack? Explain them.
5) Explain about DDoS attacks. How zombies can be used to create DDos?
6) Explain the practical side of attacks in detail.
7) State how cookies can be harmful to system?
8) Compare security services and service mechanisms.
9) Explain about "Challenge-Response Authentication" mechanism.
10) What are the principles of security? Explain.
11) What is integrity check? How can it be implemented? Explain.
12) Define salting process.
13) What is Skey? How can it be used as a security measure?
14) Explain how authentication service can be implemented? List the protocols used for authentication.
15) Define Hash Function briefly.
16) Explain Denial of Service attack? How can this attack be carried out on a internet enabled networked system?
17) What are the differences between "Packet sniffing" and "Packet spoofing"?
18) You are assigned a job of system administrator of a company. Explain how will you try to secure your own network from the following agents that pose a threat to the security:
    1) Worms
    2) Hackers
    3) Cookies
    4) Employees

# 2

# SECRET KEY CRYPTOGRAPHY

**In this unit, we are going to learn about the following topics:**

Algorithm Types
Block encryption
- ✓ Stream Vs Block Encryption

Algorithm Modes
- ECB
- CBC
- CFB
- OFB

DES Algorithm
- Double and Triple DES
- Meet-in-Middle

IDEA Algorithm

## 1. Algorithm Types and Modes

An algorithm defines what size of plain text should be encrypted in each step of the algorithm. The algorithm mode defines the details of the cryptographic algorithm, once the type is decided.

### 1.1. Algorithm Types

We have been talking about the transformation of plain text message into cipher text messages. Regardless of the techniques used, at a broad level, the generation of cipher text from plain text itself can be done in two basic ways, stream ciphers and block ciphers. This is shown in Fig. 2.**1.**

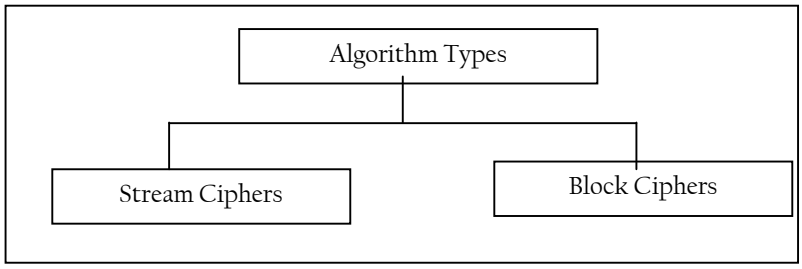



**Fig.2.1 Type of Cipher**

### 1.1.1. Stream Ciphers

In Stream Ciphers, the plain text is encrypted one bit at a time. Suppose the original message (plain text) is pay 100 in ASCII (i.e. text format). When we convert these ASCII characters to their binary values, let us assume that it translate to 01011100 (hypothetically, just for simplicity, in reality, the binary text would be much larger as each text character takes seven bits). Suppose the key to be applied is **10010101** in binary. Let us also assume that we apply the XOR logic as the encryption algorithm. XOR is quite simple to understand. As shown in fig. 2.2 in simple terms, XOR produces an output of 1 only if one input is 0 and the other is 1. The output is 0 if both the inputs are 0 or if both the inputs are 1 (hence the name exclusive).

| Input 1 | Input 2 | Input 3 |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

**Fig.2.2 Functioning of XOR logic**

We can see the effect of XOR in fig. 2.3.

As a result of applying one bit of key for every respective bit of the original message, the cipher text is generated as 11001001 in binary (ZTU91 ^% in text). Note that each bit of the plain text is encrypted one after the other. Thus, what is transmitted is 11001001 in binary, which even when translated back to ASCII would mean ZTU91 ^%. This makes no sense to an attacker, and thus protects the information.

**Note:**
Stream Cipher technique involves the encryption of one plain text bit at a time. The decryption also happens one bit at a time.

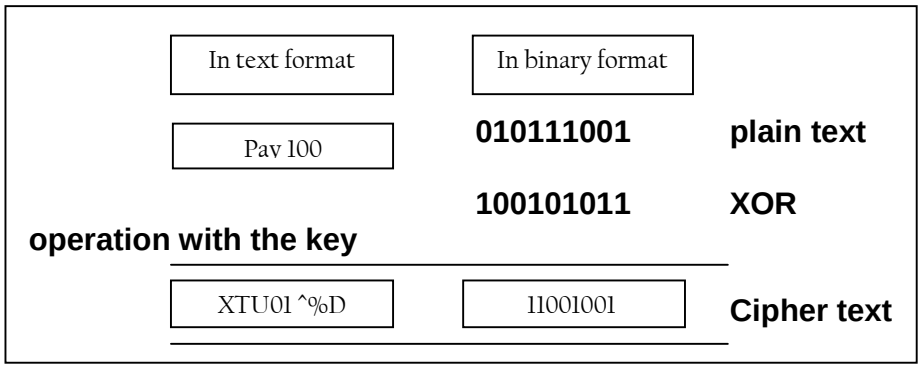



**Fig. 2.3 Stream Ciphers**

Another interesting property of XOR is that when used twice, it produces the original data. For example, suppose we have two binary numbers A=101 and B=110. We now want to perform an XOR operation on A and B to produce a third number C, i.e.:

C = A XOR B

Thus, we will have: C = 101 XOR 110

= 011

Now, if we perform C XOR A, we will get B. That is:

B = 011 XOR 101

=110

Similarly, if we perform C XOR B, we will get A. That is:

A = 011 XOR 110

= 101

This reversibility of XOR operations has many implications in cryptographic algorithms, as we shall notice.**Note: XOR is reversible** –when used twice, it produces the original values. This is useful in cryptography.

### 1.1.2. Block Ciphers

In Block Ciphers, rather than encrypting one bit at a time, a block of bits is encrypted at one go. Suppose we have a plain text FOUR_AND_FOUR that needs to be encrypted. Using block cipher, FOUR could be encrypted first, followed by _AND_ and finally FOUR. Thus, one block of characters gets encrypted at a time. During decryption, each block would be translated back to the original form. In actual practice, the communication takes place only in bits. Therefore, FOUR actually means binary equivalent of the ASCII characters FOUR. After any algorithm encrypts these, the resultant bits are converted back into their ASCII equivalents. Therefore, we get funny symbols such as Vfa%, etc. In actual practice, their binary equivalents are received, which are decrypted back into binary equivalent of ASCII FOUR. This is shown in 2.4.

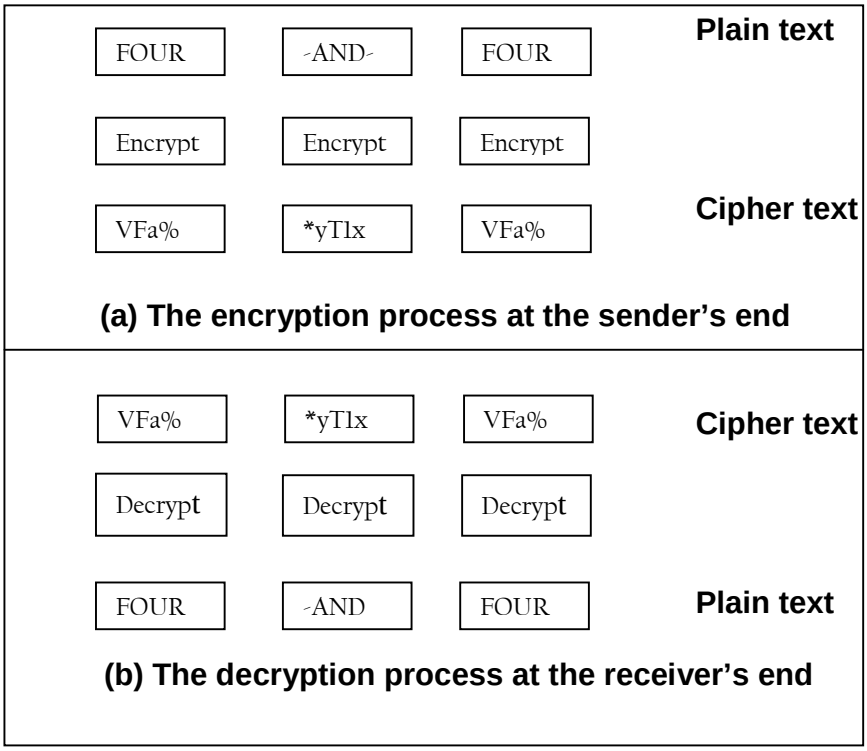



**Fig.2.4 Block Cipher**

An obvious problem with block ciphers is repeating text. For repeating text patterns, the same cipher is generated. Therefore, it could give a clue to the cryptanalyst regarding the original plain text. The cryptanalyst can look for repeating strings and try to break them. If succeeds in doing so, there is a danger that a relatively large portion of the original message is broken into, and therefore, the entire message can then be revealed with more effort. Even if the cryptanalyst cannot guess the remaining words, suppose she changes all debit to credit and vice versa in a funds transfer message, it could cause havoc! To deal with this problem, block ciphers are used in chaining mode, as we shall study later. In this approach, the previous block of cipher text is mixed with the current block, so as to obscure the cipher text, thus avoiding repeated patterns of blocks with the same contents.

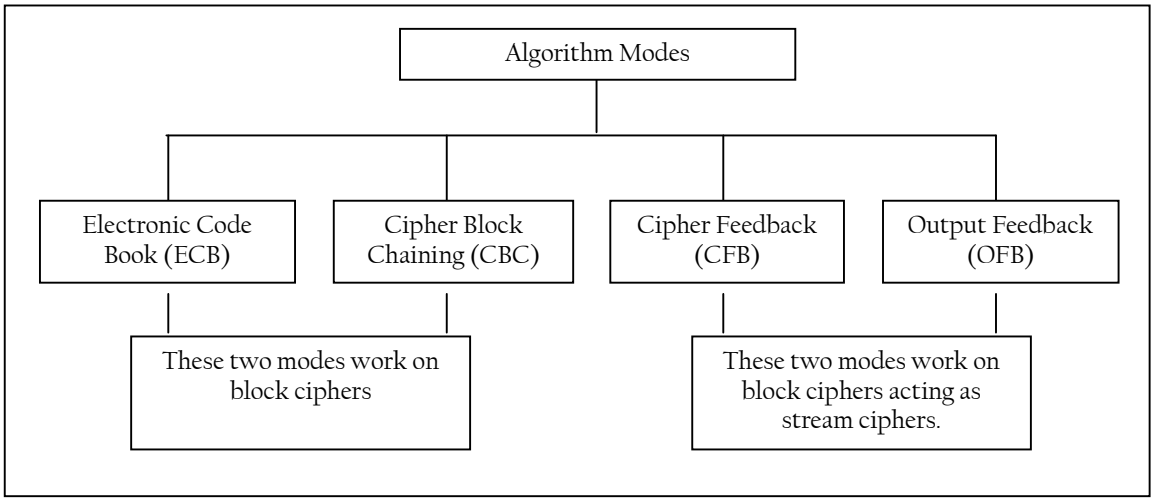
**Note:**

Block Cipher technique involves encryption of one block of text at a time. Decryption also takes one block of encrypted text at a time. Practically the blocks use in the block cipher generally contains 64 bits or more. As we have seen, stream ciphers encrypt one bit at a time. This can be very time consuming and actually unnecessary in real life. That is why block ciphers are used more

often in computer based cryptographic algorithms as compared to stream ciphers. Consequently, we will focus our attention on block ciphers with reference to algorithm modes. However, as we shall see, two of the block cipher algorithm modes can also be implemented as stream cipher modes.

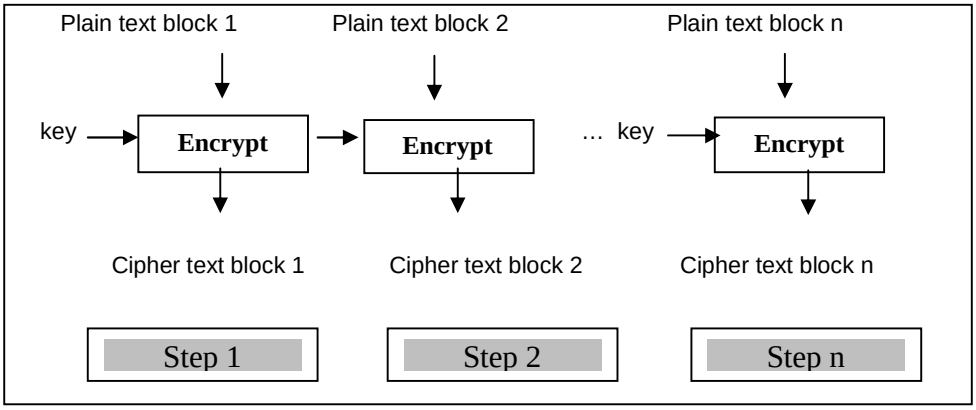
## 3. Algorithm Modes:

An algorithm mode is a combination of a series of the basic algorithm steps on block cipher, and some kind of feedback from the previous step. There are four important algorithm modes, namely, Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB) and Output Feedback (OFB). This is shown in the following figure. The first two modes operate on block cipher, whereas the latter two modes are "block-cipher" modes, which can be used as if they are working on stream cipher.

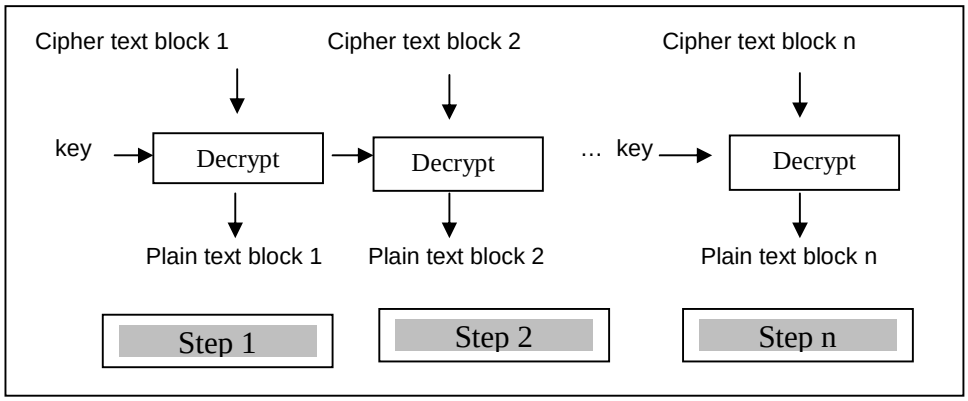



### 1. Electronic Code Book (ECB) mode:

Electronic Code Book (ECB) is the simplest mode of operation. Here the meaning plain text message is divided into blocks of 64 bits each. Each such block is encrypted independently of the other blocks. For all blocks in a message, the same key is used for encryption. This process is shown in fig. 2.5




At the receiver's end, the incoming data is divided into 64-bit blocks, and by using the same key as was used for encryption, each block is decrypted to produce the corresponding plain text block. This process is shown in fig. 2.6




In ECB, since a single key is used for encrypting all the blocks of a message, if a plain text block repeats in the original message the corresponding cipher text block will also repeat in the encrypted message. Therefore, ECB is suitable only for encrypting small messages, where the scope for repeating the same plain text blocks is quite less.

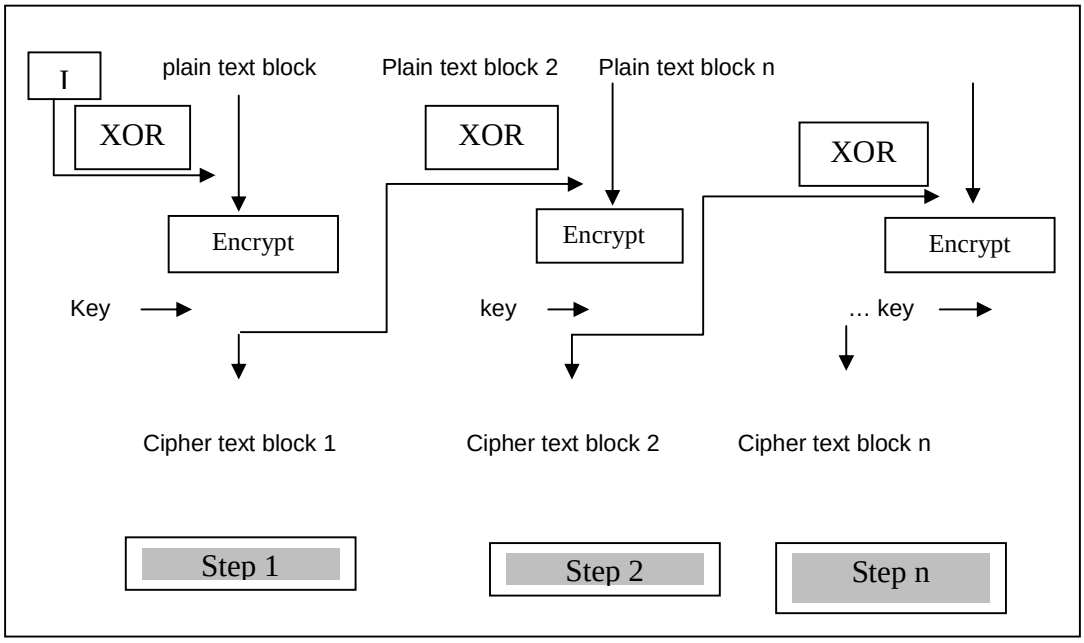
## 2. Cipher Block Chaining (CBC) mode:

We saw that in the case of ECB, within a given message (i.e. for a given key), a plain text block always produces the same cipher text block. Thus, if a block of plain text occurs more than once in the input, the corresponding cipher text block will also occur more than once in the output, thus providing some clues to a cryptanalyst. To overcome this problem, Cipher Block Chaining (CBC) mode ensures that even if a block of plain text repeats in the input, these two (or more) identical plain text blocks yield totally

different cipher text blocks in the output. For this, a feedback mechanism is used, as we shall learn now.

Chaining adds a feedback mechanism to a block cipher. In Cipher Block Chaining (CBC), the results of the encryption of the previous block are fed back into the encryption of the current block. That is, each block is used to modify the encryption of the next block. Thus each block of cipher text is dependent on the corresponding current input plain text block, as well as all the previous plain text blocks.

The encryption process of CBC is depicted in fig. 2.7 and described thereafter.




1. As shown in the figure, the first step receives two inputs: the first block of plain text and a random block of text, called as Initialization Vector (IV).

    a. The IV has no special meaning: it is simply used to make each message unique. Since the value of IV is randomly generated, the likelihood of it repeating in two different messages is quite rare. Consequently, IV helps in making the cipher text somewhat unique or at least quite different from all the other cipher texts in a different message. Interestingly, it is not mandatory to keep IV secret –it can be known to everybody. This seems slightly concerning and confusing. However, if we relook at the operation of CBC, we will realize that IV is simply one of the two inputs to the first encryption step. The output of step 1 is cipher text block 1, which is also one of the two inputs to the second encryption

step. In other words, Cipher text block 1 is also an IV for step 2! Similarly, Cipher text block 2 is also an IV for step 3, and so on. Since all these cipher text blocks will be sent to the receiver, we are actually anyway sending all IVs for step 2 onwards! Thus, there is no special reason why the IV for step 1 should be kept secret. The key used for encryption is what needs to be kept secret. However, in practice, for maximum security, both the key and the IV are kept secret.

b. The first block of cipher text and IV are combined using XOR and then encrypted using a key to produce the first cipher text block. The first cipher text block is then provided as a feedback to the next plain text block, as explained below.

2. In the second step, the second plain text block is XORed with the output of step 1, i.e. the first cipher text block. It is then encrypted with the same key, as used in step 1. This produces cipher text block 2.

3. In the third step, the third plain text block is XORed with the output of step 2, i.e. the second cipher text block. It is then encrypted with the same key, as used in step 1.

4. This process continues for all the remaining plain text blocks of the original message.

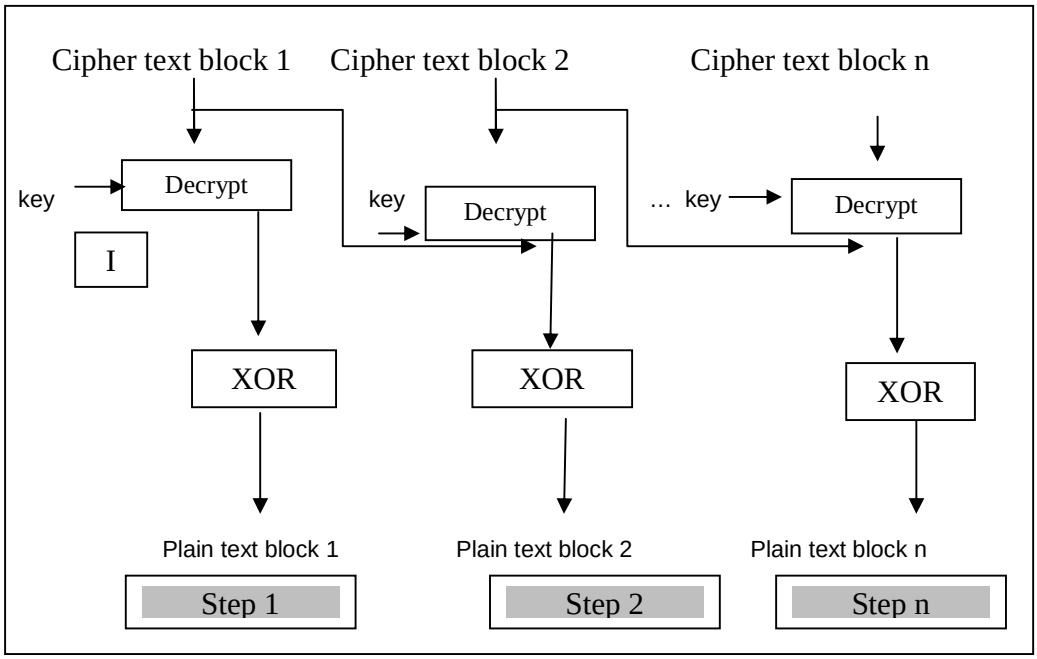
Remember that the IV is used only in the first plain text block. However, the same key is use for encryption of all plain text blocks.

1. The Cipher text block 1 is passed through the decryption algorithm using the same key, which was used during the encryption process for all the plain text blocks. The output of this step is then XORed with the IV. This process yields Plain text block 1.

2. In step 2, the Cipher text block 2 is decrypted, and its output is XORed with Cipher text block 1, which yields Plain text block 2.

3. This process continues for all the Cipher text blocks in the encrypted message.

The decryption process is shown in fig. 2.8

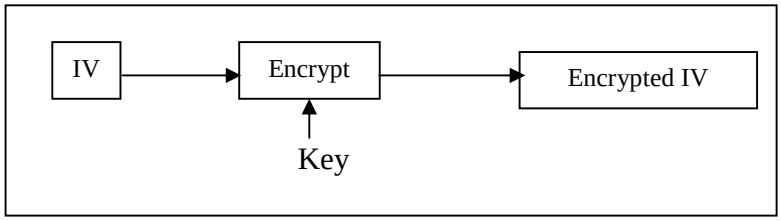



**3.Cipher Feedback (CFB) mode:**

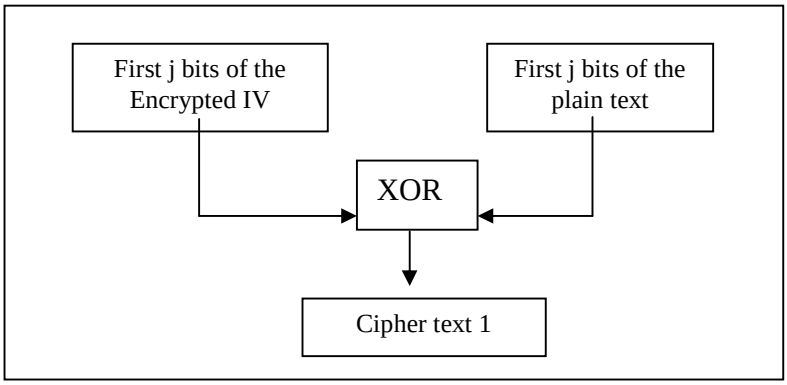
Not all applications can work with blocks of data. Security is also required in applications that are character-oriented. For instance, an operator can be typing keystrokes at a terminal, which need to be immediately transmitted across the communications link in a secure manner, i.e. by using encryption. In such situations, stream cipher must be used. The Cipher Feedback (CFB) mode is useful in such cases. In this mode, data is encrypted in units that are smaller (e.g. they could be of size 8 bits, i.e. the size of character typed by an operator) than a defined block size (which is usually 64 bits).

Let us understand how CFB mode works, assuming that we are dealing with j bits at a time (as we have seen, usually, but not always, j=8). Since CFB is slightly more complicated as compared to the first two cryptography modes, we shall study CFB in a **step-by-step fashion.**

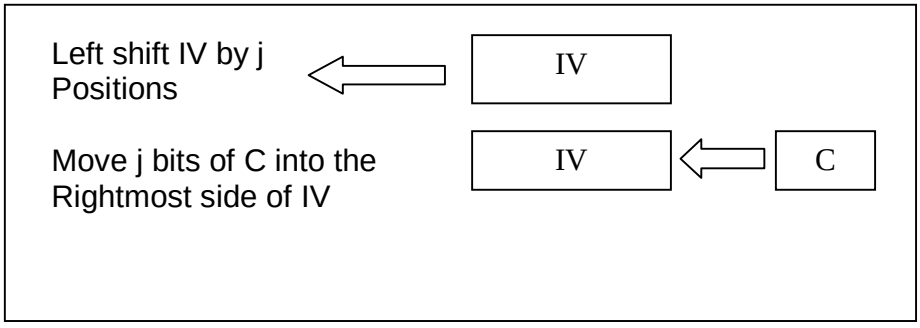
**Step 1:** Like CBC, a 64-bit Initialization vector (IV) is used in the case of CFB mode. The IV is kept in a shift register. It is encrypted in the first step to produce a corresponding 64-bit IV cipher text. This is **shown fig. 2.9.**




**Step 2**: Now, the leftmost (i.e. the most significant) j bits of the encrypted IV are XORed with the first j bits of the plain text. This produces the first portion of cipher text (say C) as shown in fig. 1.5. C is transmitted to the receiver.



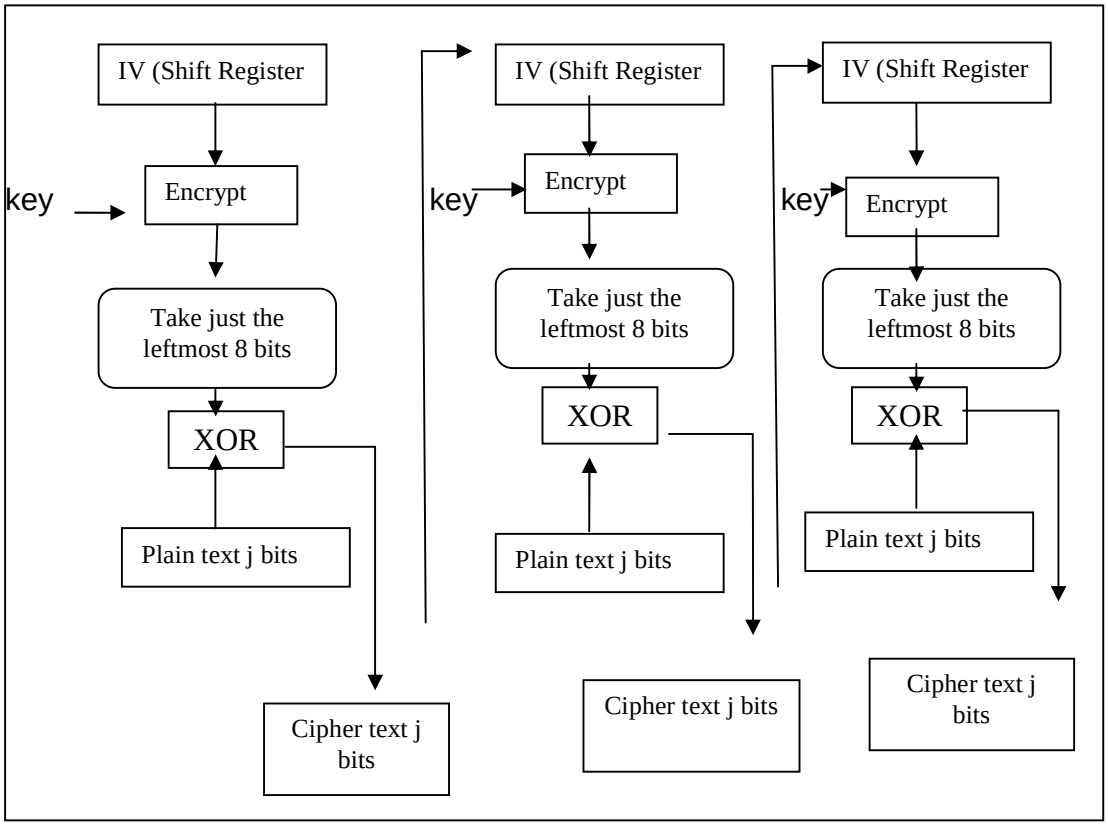

**Step 3:** Now, the bits of IV (i.e. the contents of the shift register containing IV) are shifted left by j positions. Thus the rightmost j positions of the shift register now contain unpredictable data. These rightmost j positions are now filled with C. This is shown in fig. 2.10.




**Step 4**: Now step 1 through 3 continue until all the plain text units are encrypted. That is, the following steps repeat:

- IV is encrypted.
- The leftmost j bits resulting from this encryption process are XORed with the next j bits of the plain text.
- The resulting cipher text portion (i.e. comprising of the next j bits of the cipher text) is sent to the receiver.

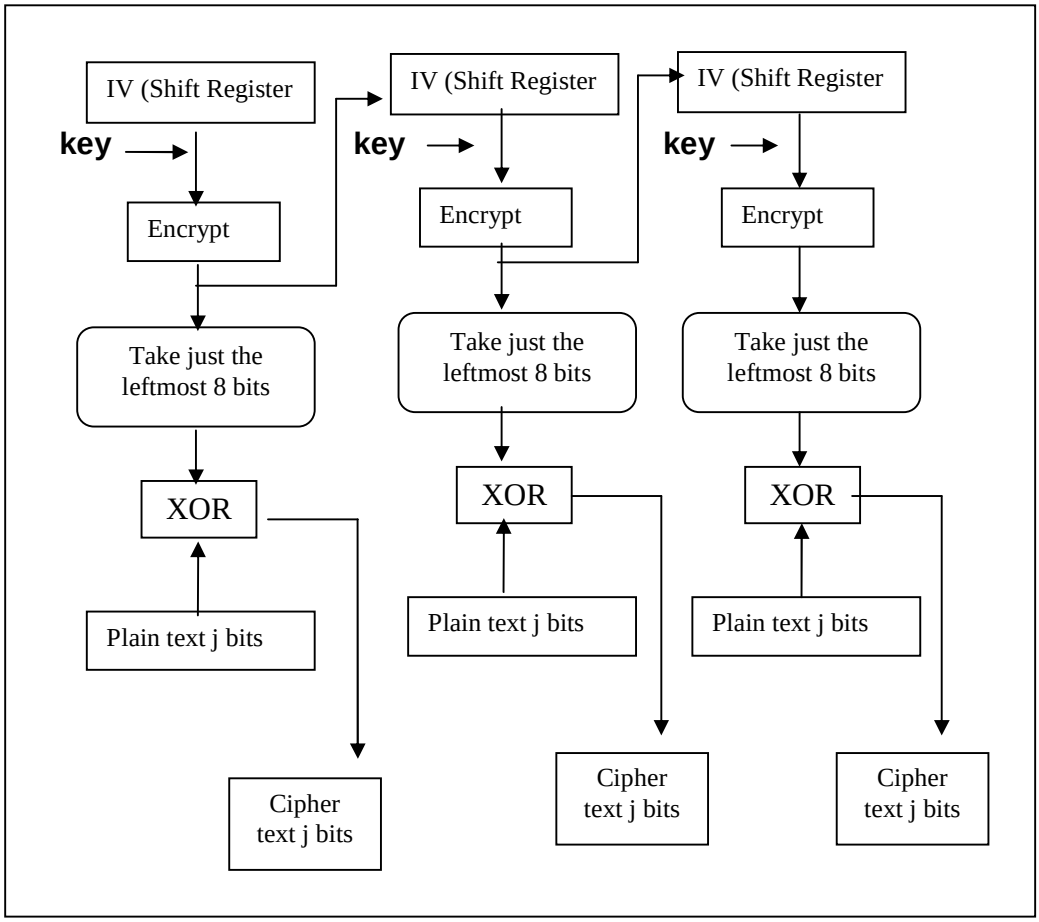
- The shift register containing the IV is left-shifted containing the IV.
- Fig. 2.11 shows the overall conceptual view of the CFB mode.




At the receiver's end, the decryption process is pretty similar, with minor changes.

**4.Output Feedback (OFB) mode:**

The Output Feedback (OFB) mode is extremely similar to the CFB. The only difference is that in case of CFB, the cipher text is fed into the next stage of encryption process. But in the case of OFB, the output of the IV encryption process is fed into the next stage of encryption process. Simply draw the block diagram of the OFB process, as shown in fig. 2.11.




**DES (Data Encryption Standard) Cipher Algorithm**
**DES Cipher** - A 16-round Festal cipher with block size of 64 bits. DES stands for Data Encryption Standard. IBM developed DES in 1974 in response to a federal government public invitation for data encryption algorithms. In 977, DES was published as a federal standard, FIPS PUB 46.

**Algorithm:**

**Step 1:** 64 bit plain text blocks are handed over to the initial permutation (IP) function.

**Step 2:** IP is performed on the plain text.

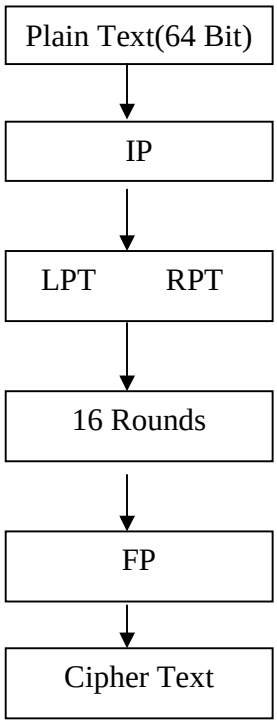
**Step 3:** IP produces 2 halves; say LPT and RPT, both of 32 bit each.

**Step 4:** Perform 16 rounds of encryption process each with its own key.
Rounds are defined as follows in the algorithm:
4a: Key transformation 4b: Expansion Permutation (EP)
4c: S-Box Substitution

4d: P-Box Permutation 4e: XOR and Swap.

**Step 5:** LPT and RPT are rejoined finally and a Final Permutation (FP) is performed on the combined block. Step 6: The result of this process produces 64-bit cipher text.

**Diagrammatical representation:**




**Explanation of the algorithm:**

- Comparing the IP table performs IP. It happens only once, and it happens before the first round. It suggests how the transposition in IP should proceed, as shown in the IP table.

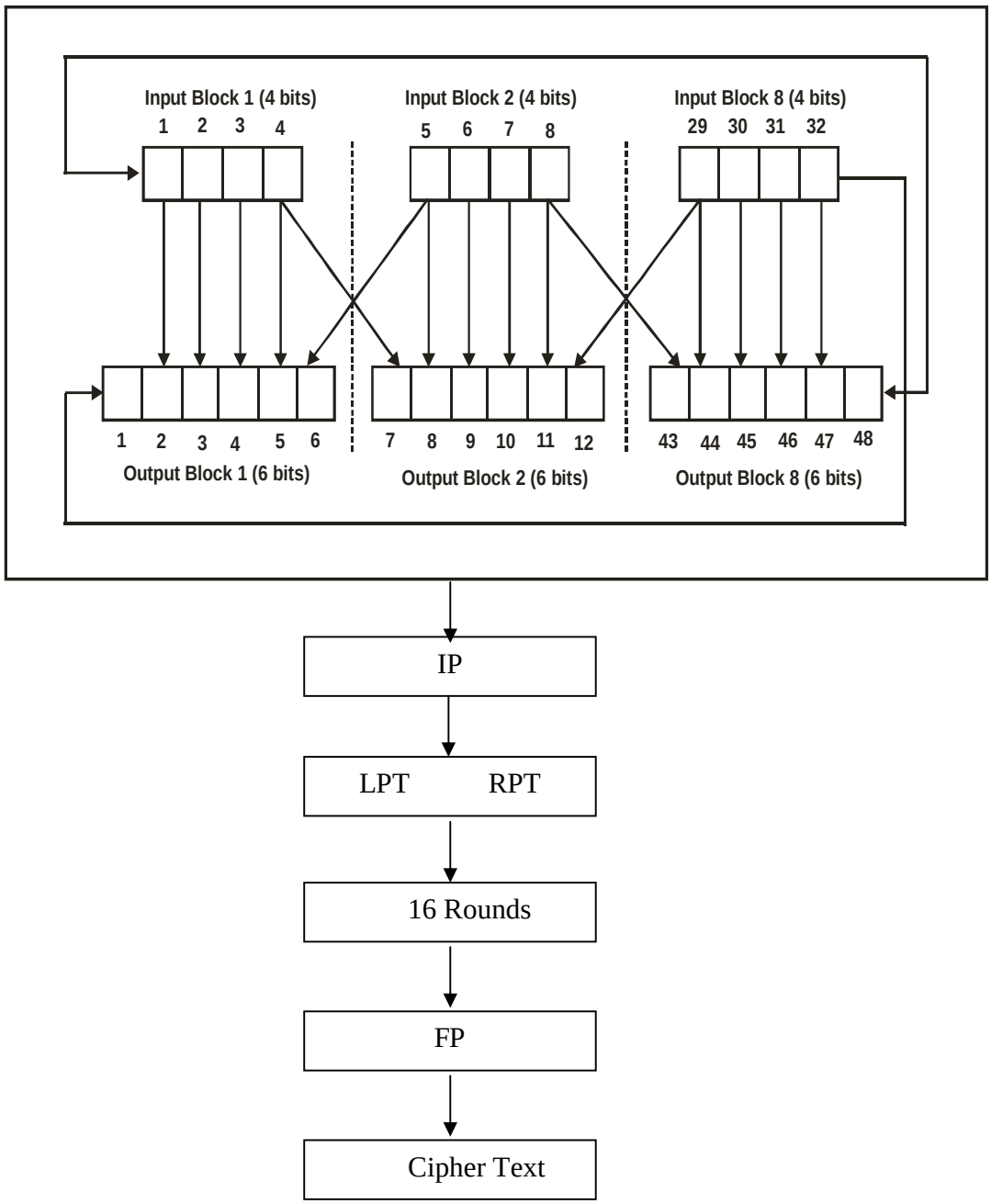
- In the rounds, **step 1 is key transformation.**
  That is achieved by
  1. Shifting the key position by considering the Round Table.
  2. Compare the Compression Table to get the sub key of 48 bits.

**Step 2 is Expansion Permutation (EP).**

In this step, the 32-bit RPT is expanded to 48 bits as it of key length. The process is shown as under:

The 32-bit text is divided into 8 blocks of 4 bits each. Then by adding 2 bits extra, that is the first bit of the block 1 is the last bit of the block 8 and the last bit of the block 8 is the first bit of the $7^{th}$ block the 48-bit text is obtained.

Diagram for the same is as below:




After this expansion it will be compared with the Expansion Permutation Table.

**Step 3 in Round is S-Box Substitution.**

1. This step reduces 48 bits RPT into 32 bits because LPT is of 32 bits.

2. It accepts 48 bits, does some XOR logic and gives 32 bits.

- The 48 bits key (Result of Step 1) and the 48 bits of RPT (Result of Step 2) will be XOR and the output will be 48 bits Input block and that will be given as the input for the S-Box Substitution.

- The 48-bit block text will be divided into 8 blocks of 6 bits each.

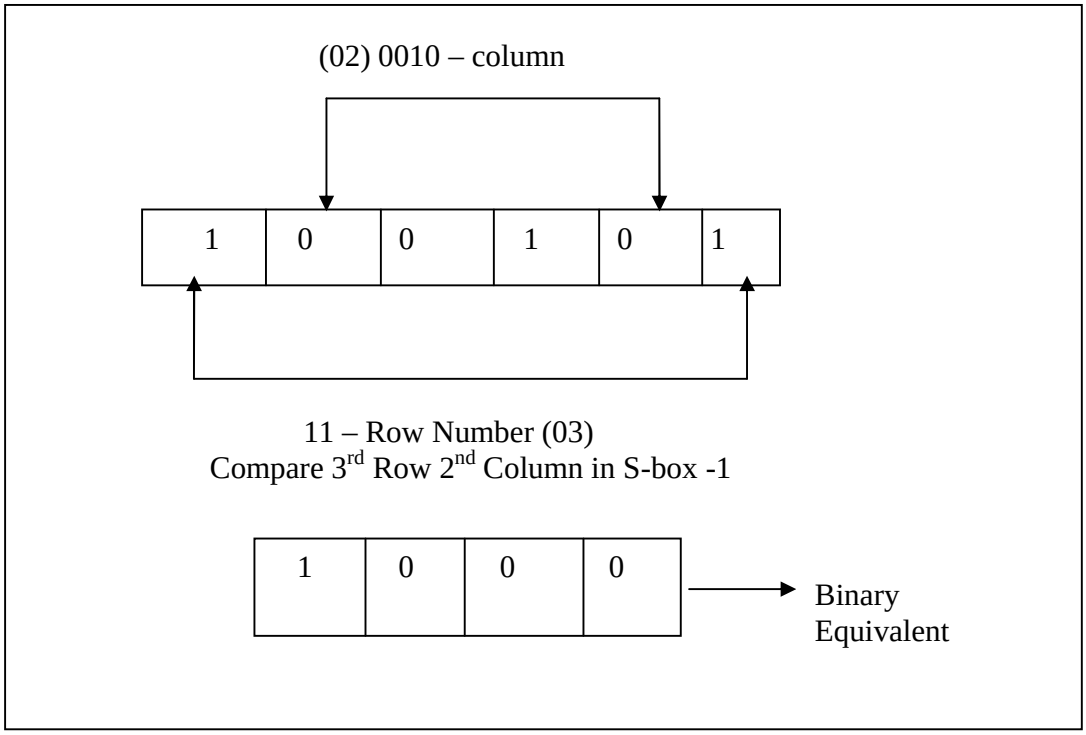
- Decimal equivalent of the first and last bit in a block denotes the row number and decimal equivalent of the bit 2,3,4 and 5 denotes the column number of the S-Box Substitution table.

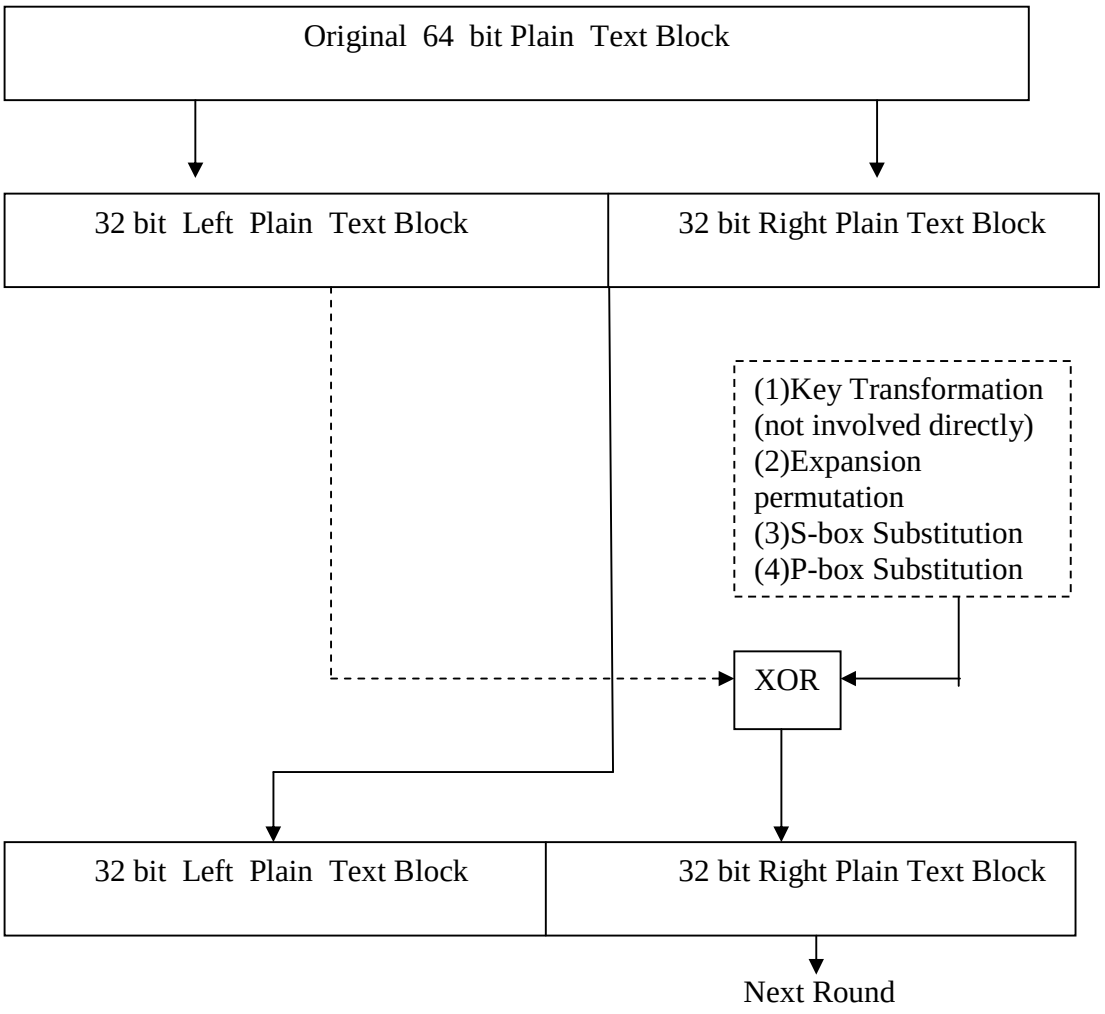
- Check the value and take the binary equivalent of the number.

- The result is 4-bit binary number.

- For example if the 6-bit number is 100101 then the first and last bit is 11 and the decimal equivalent of the number is 3. The remaining bits are 0010 and the decimal equivalent of the number is 2. If it is the first block of input, then check the $3^{rd}$ row $2^{nd}$ column value in the Sbox-1 substitution table. It is given as 1 in the table. Binary equivalent of 1 is 0001.

- The input 100101 of 6-bit is now reduced to 0001 after S-Box Substitution.


(02) 0010 – column
1
0
0
1
0
1
11 – Row Number (03)
Compare 3rd Row 2nd Column in S-box -1
1
0
0
0
Binary
Equivalent


Original 64 bit Plain Text Block
32 bit Left Plain Text Block
32 bit Right Plain Text Block
(1)Key Transformation
(not involved directly)
(2)Expansion
permutation
(3)S-box Substitution
(4)P-box Substitution
XOR
32 bit Left Plain Text Block
32 bit Right Plain Text Block
Next Round

**Step 4 in Round is P-Box Permutation.**

In this step, the output of S-Box, that is 32 bits are permuted using a p-box. This mechanism involves simple permutation that is replacement of each bit with another bit as specified in the p-Box table, without any expansion or compression. This is called as P-Box Permutation. The P-Box is shown below.

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

For example, a 16 in the first block indicates that the bit at position 16 moves to bit at position 1 in the output.

**Step 5 is XOR and Swap**

The untouched LPT, which is of 32 bits, is XORed with the resultant RPT that is with the output produced by P-Box permutation. The result of this XOR operation becomes the new right half. The old right half becomes the new left half in the process of swapping.

At the end of 16 rounds, the Final Permutation is performed only once. This is a simple transposition based on the Final Permutation Table.

- The output of the Final permutation is the 64-bit encrypted block.

**4. Multiple Encryptions with DES:**

**4.1. Improving the Security of DES**

You can improve the security of DES by performing multiple encryptions, known as **super encryption.** The two most common ways of doing this are with double encryption (**Double** DES*)* and with triple encryption (**Triple** DES).

While double DES appears to add significant security, research has found some points of attack, and therefore experts recommend Triple DES for applications where single DES is not adequate.

**1. Double DES:**

In Double DES, each 64-bit block of data is encrypted twice with the DES algorithm, first with one key, then with another, as follows:

1. **Encrypt with (key 1).**
2. **Encrypt with (key 2).**

**Plaintext → (key1) → (key2) → ciphertext:**

Double DES is not significantly more secure than single DES. In 1981, Ralph Merkle and Martin Hellman published an article in which they outlined a so-called "meet-in-the-middle attack. The meet-in-the-middle attack is a **known plaintext attack** which requires that an attacker have both a known piece of plaintext and a block of that same text that has been encrypted. (These pieces are surprisingly easily to get.) The attack requires storing $\mathbf{2^{56}}$ intermediate results when trying to crack a message that has been encrypted with DES (a total of $\mathbf{2^{59}}$ **bytes**), but it reduces the number of different keys you need to check from $\mathbf{2^{112}}$ **to** $\mathbf{2^{57}}$. "This is still considerably more memory storage than one could comfortably comprehend, but it's enough to convince the most paranoid of cryptographers that double encryption is not worth anything. In other words, because a message encrypted with DES can be forcibly decrypted by an attacker performing an exhaustive key search today, an attacker might also be able to forcibly decrypt a message encrypted with Double DES using a meet-in-the-middle attack at some point in the future.

**2. Triple DES:**

This is a known variant of *DES* and is very easy to implement given an implementation of **DES.** Its strength lies in the new key length of 168 bits which addresses the biggest weakness with standard **DES** albeit with an unorthodox key length like standard **DES.**

The dangers of the Merkle-Hellman meet-in-the-middle attack can be circumvented by performing three block encryption operations. This method is called Triple DES.In practice, the most common way to perform Triple DES is:

1. **Encrypt with (key1).**
2. **Decrypt with (key2).**
3. **Encrypt with (key3).**

The advantage of this technique is that it can be backward compatible with single DES, simply by setting all three keys to be the same value.

To decrypt, reverse the steps:

1. **Decrypt with (key3).**
2. **Encrypt with (key2).**
3. **Decrypt with (key1).**

For many applications, you can use the same key for both key1 and key3 without creating a significant vulnerability. Triple DES appears to be roughly as secure as single DES would be if it had a **112-bit key**. How secure is this really? Suppose you had an integrated circuit which could perform one million Triple DES encryptions per second, and you built a massive computer containing one million of these chips to forcibly try all Triple DES keys. This computer, capable of testing $\mathbf{10^{12}}$ encryptions per second, would require:

$$\mathbf{2^{112} = 5.19 \times 10^{33} \text{ encryption operations}}$$

$$\mathbf{5.19 \times 10^{33} \text{ encryption operations} / 10^{12} \text{ operations/sec} = 5.19 \times 10^{21} \text{ sec}}$$

$$\mathbf{= 1.65 \times 10^{14} \text{ years.}}$$

This is more than **16,453 times older** than the currently estimated age of the universe (approximately $\mathbf{10^{10}}$ **years**).

Apparently, barring new discoveries uncovering fundamental flaws or weaknesses with the DES algorithm, or new breakthroughs in the field of cryptanalysis, Triple DES is the most secure private key encryption algorithm that humanity will ever need.

**"Meet in the middle" example:**

Let us assume that we are given a message *M*, its encryption *C*, and **double DES** was employed, i.e. $\boldsymbol{C = Ek1\,(Ek2\,(M))}$**.** One calculates $\boldsymbol{Ek2\,[M]}$ for all ***K2*** and stores these values in a hash table. One then computes $\boldsymbol{E{-}1\ k1\ [C]}$**,** for all *k*1 looks for collisions in the hash table that can be investigated further. This approach uses time at most 257 so it is only marginally more expensive than single **DES**. On a more pessimistic note the procedure also uses 256 memory and that might be harder to come by.

**Breaking DES:**

Given a set of known plaintexts and crypto texts, it is possible to analyze the pairs and construct and reduce the number of keys it is necessary to check. Examples of such an approach are:

1. **Differential crypto-analysis.**
2. **Linear crypto-analysis.**

The former algorithm can successfully crypt-analyze *DES* by seeing 247 chosen plaintexts, the latter algorithm, however, requires ≈ 243 message blocks, which is ≈ 246 bytes, or 64 Tera bytes.

## 5. IDEA ALGORITHM AND ITS WORKING:

The IDEA in perceived as one of the strongest cryptographic algorithms. It was launched in 1990 and underwent certain changes in names and capabilities as shown in table.

| Year | Name |
|---|---|
| **1990** | **Proposed Encryption Standard(PES)** |
| **1991** | **Improved Proposed Encryption Standard(IPES)** |
| **1992** | **International Data Encryption Algorithm(IDEA)** |

One popular email privacy technology known as Pretty Good Privacy (PGP) is based on IDEA.

(1) IDEA is block cipher.

(2) IDEA is reversible like DES , i.e. the same algorithm is used for encryption and decryption.

(3) It uses both confusion and diffusion for encryption.

**Algorithm:**

(1) Consider the input plain text of 64 bits.

(2) Divide the input plain text into 4 portions each of size 16 bits (Say p1 to P4).

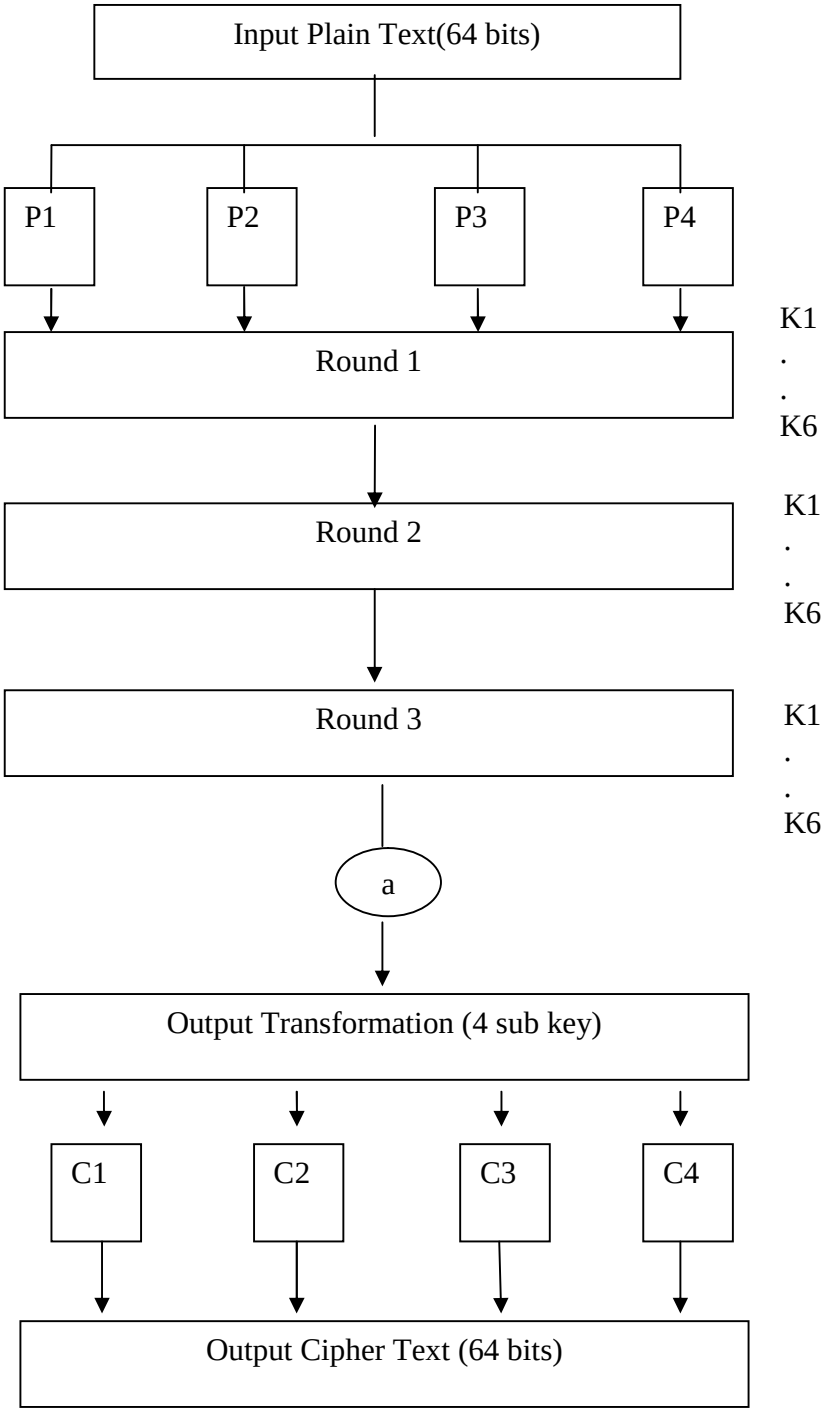
(3) Now perform 8 rounds of algorithm.

(a) In each round 6 sub-keys are generated from the original key. Each of the sub-keys consists of 16-bits. These six sub-keys are applied to four input blocks P1 to P4.Thus for first round, we have 6 keys say k1 to k6;for second round , we have k7 to k12.Finally for eight round we have keys k43 to k48.

(b)Multiply, add and XOR the plain text blocks with sub keys.

(4) Perform an output transformation in sub-keys.

(5) Combine all the 4 blocks of output transformation to get the cipertext of 64 bits.




**1. Details of first round in IDEA:**

The initial key consists of 128 bits from which 6 sub-keys k1 to k6 are generated for the first round.

Since k1 to k6 consist of 16 bits each, out of the original 128 bits, the first 96 bits(6 sub-keys * 16 bits per sub – key) are used for the first round. Thus at the end of the first round, bits 97-128 of the original keys are unused.

**2. Details of second round in IDEA:**

In 2nd round 31 unused bits are used. For second round we still require(96-31=65) more bits.But the original key 128 bits are exhausted.

Now IDEA uses the techniques of key shifting. At this stage the original key is shifted left circularly by 25 bits that is , the 26th bit of original key moves to the first position and becomes the first bit after the shift, and the 25th bit of the original key moves to the last position and becomes the 128th bit after the shift

**3. Details of one round in IDEA:**

1. Multiply P1 and k1
2. Add P2 and K2
3. Add P3 and k3
4. Multiply P4 and k4
5. XOR the results of steps 1 and 3.
6. XOR the results of steps 2 and 4.
7. Multiply step 5 and k5
8. Add steps 6 and step 7.
9. Multiply the result of step 8 and k6.
10. Add steps 7 and 9.
11. XOR the results of steps 1 and 9.
12. XOR the results of steps 3 and 9.
13. XOR the results of steps 2 and 10.
14. XOR the results of steps 4 and 10.

**4. Details of output Transformation:**

(1)The output transformation is a onetime operation. It takes place at the end of the 8th round.

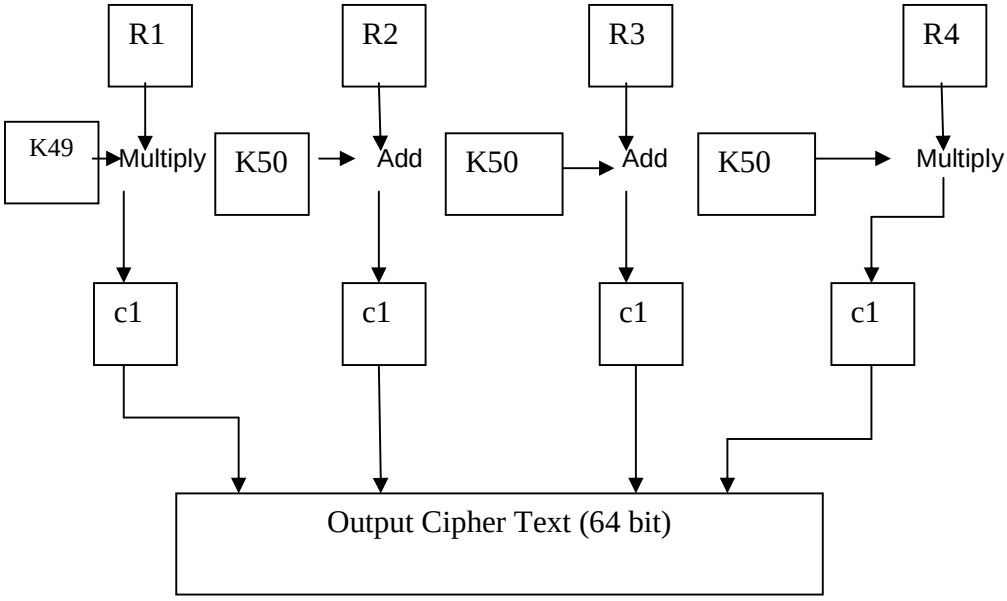
(2)It is 64 bit value divided into 4 sub-blocks (say R1 to R4 each consisting of 16 bits).

Step 1: Multiply R1 and k49.

Step2: Add R2 and k50.

Step3: Add r3 and k51.

Step 4: Multiply r4 and k52.




**A Symmetric Cryptosystems Comparison Table**

| Cipher | Security | Speed (486 pc) | Key length |
|---|---|---|---|
| **DES** | low | 400 kb/s | 56 bits |
| **Triple DES** | good | 150 kb/s | 112 bits |
| **IDEA** | good* | 200 kb/s | 128 bits |
| **Triple IDEA** | very good* | ~100 kb/s | 256 bits |

*** The algorithm is believed to be strong**
**** The algorithm itself is good, but it has a built-in weakness**

**Questions for practice:**

1) Distinguish between block and stream cipher.
2) Explain the algorithm modes in detail. Compare the four different modes.
3) Explain DES, one of the symmetric cryptosystem algorithms in detail.
4) Explain IDEA and its working in detail.
5) What are the variations of DES? How they can improve the security?
6) Discuss the problems with DES and its variations. Give example.
7) Compare symmetric cryptosystems and explain the features.

# 3

# HASH FUNCTIONS AND MESSAGE DIGESTS

**In this unit, we are going to learn about the following topics:**

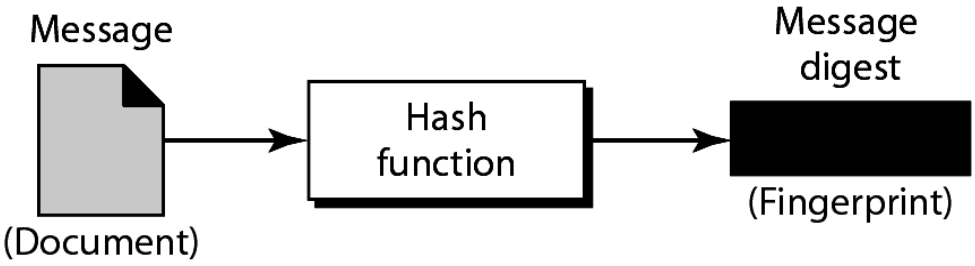
- Meaning of Hash
- Requirements of Hash
- Idea behind Hash
- MD2 algorithm

***MD4 algorithm***

- MD5 algorithm
- SHS
  - ✓ MAC
  - ✓ HMAC

### 3.1. Signing the Digest:

We said before that public-key encryption is efficient if the message is short. Using a public key to sign the entire message is very inefficient if the message is very long. The solution is to let the sender sign a digest of the document instead of the whole document. The sender creates a miniature version or digests of the document and signs it; the receiver then checks the signature on the miniature. To create a digest of the message, we use a hash function. The hash function creates a fixed-size digest from a variable-length message, as shown in Figure.



## 1. Requirement of a message digest

We can summarize the requirements of the message digest concept, as follows:

- Given a message, it should be very easy to find its corresponding message digest. Also for a given message, the message digest must always be the same.

- Given a message digest, it should be very difficult to find the original message for which the digest was created.
- Given any two messages, if we calculate their message digests, the two message digests must be different.

Another basis of message digest is that it should not give any clue or indication of the original message. i.e. it should not be possible to revert back to original message from the digest. Also, for a given message it's digest should be the same always. Different algorithms are used to convert original message into its message digest.

## 2. Idea of a Message Digest.

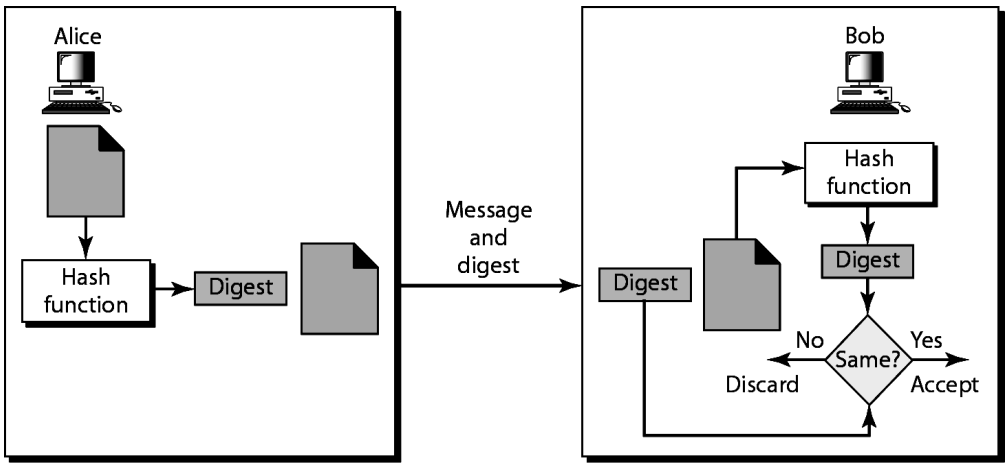
The concept of message digests is based on similar principles. However, it is slightly wider in scope. For instance, suppose that we have a number 4000 and we divide it by 4 to get 1000 Thus, 4 can become a fingerprint of the number 4000. Dividing 4000 by 4 will always yield 1000. If we change either 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. 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 digits of the multiplication operation, if the result is a two-digit number. Thus, we perform a hashing operation (or a message digest algorithm) over a block of data to produce its hash or message digest, which is smaller in size than the original message. Actually, the message digests are not so small and straightforward to compute. Message digests usually consist of 128 or more bits. This

means that the chance of any two-message digests being the same is anything between 0 and at least 2128. The message digest length is chosen to be so long with a purpose. This minimizes that the scope for two messages digests being the same.

After the digest has been created; it is encrypted (signed) using the sender's private key. The encrypted digest is attached to the original message and sent to the receiver.




The popularly used ones are MD5 or Message Digest 5 (developed by Rivest) a modified version of earlier MD4, MD3 and MD2, while the first one was simply MD, and the SHA (Secure Hash Algorithm) developed by National Institute of Standards and Technology (NISI) in 1993. SHA-I is promoted & prominently used than the MD5 algorithm.

Among several functions are called MD5 (Message Digest 5) and SHA-I (Secure Hash Algorithm I). The first one produces a 120-bit digest. The second produces a 160-bit digest.

Note that a hash function must have two properties to guarantee its success. First**, hashing is one-way**; the digest can only be created from the message, not vice versa. Second, **hashing is a one-to-one function**; there is little probability that two messages will create the same digest.

### 3.2 MD2 algorithm:

The algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD2 algorithm is intended for digital signature

applications, where a large file must be "compressed" in a secure manner before being signed with a private (secret) key under a public-key cryptosystem such as RSA. The reference implementation of MD2 is more portable.

## 1. Terminology and Notation

In this document, a "byte" is an eight-bit quantity. Let x_i denote "x sub i". If the subscript is an expression, we surround it in braces, as in x_{i+1}. Similarly, we use ^ for superscripts (exponentiation), so that x^i denotes x to the i-th power. Let X xor Y denote the bit-wise XOR of X and Y.

## 2. MD2 Algorithm Description

We begin by supposing that we have a b-byte message as input, and that we wish to find its message digest. Here b is an arbitrary nonnegative integer; b may be zero, and it may be arbitrarily large. We imagine the bytes of the message written down as follows:

**m_0 m_1 ... m_{b-1}**

The following five steps are performed to compute the message digest of the message.

### Step 1. Append Padding Bytes

The message is "padded" (extended) so that its length (in bytes) is congruent to 0, modulo 16. That is, the message is extended so that it is a multiple of 16 bytes long. Padding is always performed, even if the length of the message is already congruent to 0, modulo 16. Padding is performed as follows: "i" bytes of value "i" are appended to the message so that the length in bytes of the padded message becomes congruent to 0, modulo 16. At least one byte and at most 16 bytes are appended. At this point the resulting message (after padding with bytes) has a length that is an exact multiple of 16 bytes.

Let M [0 ... N-1] denote the bytes of the resulting message, where N is a multiple of 16.

### Step 2. Append Checksum

A 16-byte checksum of the message is appended to the result of the previous step. This step uses a 256-byte "random" permutation constructed from the digits of pi. Let S[i] denote the i-th element of this table. Do the following:

```
/* Clear checksum. */
  For i = 0 to 15 do:
    Set C[i] to 0.
  end /* of loop on i */
  Set L to 0.
 /* Process each 16-word block. */
  For i = 0 to N/16-1 do

    /* Checksum block i. */
    For j = 0 to 15 do
      Set c to M[i*16+j].
      Set C[j] to S[c xor L].
      Set L to C[j].
     end /* of loop on j */
   end /* of loop on i */
```

The 16-byte checksum C[0 ... 15] is appended to the message. Let M[0] with checksum), where N' = N + 16.

**Step 3. Initialize MD Buffer**

A 48-byte buffer X is used to compute the message digest. The buffer is initialized to zero.

**Step 4. Process Message in 16-Byte Blocks**

This step uses the same 256-byte permutation S as step 2 does.

Check out these exercises.

```
/* Process each 16-word block. */
  For i = 0 to N'/16-1 do
    /* Copy block i into X. */
    For j = 0 to 15 do
      Set X[16+j] to M[i*16+j].
      Set X[32+j] to (X[16+j] xor X[j]).
     end /* of loop on j */
    Set t to 0.
```

```
    /* Do 18 rounds. */
    For j = 0 to 17 do
      /* Round j. */
      For k = 0 to 47 do
        Set t and X[k] to (X[k] xor S[t]).
      end /* of loop on k */
      Set t to (t+j) modulo 256.
    end /* of loop on j */
  end /* of loop on i */
```

**Step 5. Output**

The message digest produced as output is X[0 ... 15]. That is, we begin with X[0], and end with X[15].
This completes the description of MD2.

**Summary**

The MD2 message-digest algorithm is simple to implement, and provides a "fingerprint" or message digest of a message of arbitrary length. It is conjectured that the difficulty of coming up with two messages having the same message digest is on the order of 2^64 operations, and that the difficulty of coming up with any message having a given message digest is on the order of 2^128 operations. The MD2 algorithm has been carefully scrutinized for weaknesses. It is, however, a relatively new algorithm and further security analysis is of course.

**3..3. MD4 algorithm:**

**Executive Summary**

The algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. It is conjectured that it is computationally infeasible to produce two messages having the same message digest, or to produce any message having a given prespecified target message digest. The MD4 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA.

The MD4 algorithm is designed to be quite fast on 32-bit machines. In addition, the MD4 algorithm does not require any large substitution tables; the algorithm can be coded quite compactly. The MD4 algorithm is being placed in the public domain for review and possible adoption as a standard.

## 1. Terminology and Notation:

In this document a "word" is a 32-bit quantity and a "byte" is an eight-bit quantity. A sequence of bits can be interpreted in a natural manner as a sequence of bytes, where each consecutive group of eight bits is interpreted as a byte with the high-order (most significant) bit of each byte listed first. Similarly, a sequence of bytes can be interpreted as a sequence of 32-bit words, where each consecutive group of four bytes is interpreted as a word with the low-order (least significant) byte given first.

Let x_i denote "x sub i". If the subscript is an expression, we surround it in braces, as in x_{i+1}. Similarly, we use ^ for superscripts (exponentiation), so that x^i denotes x to the i-th power.

Let the symbol "+" denote addition of words (i.e., modulo-2^32 addition). Let X <<< s denote the 32-bit value obtained by circularly shifting (rotating) X left by s bit positions. Let not(X) denote the bit-wise complement of X, and let X v Y denote the bit-wise OR of X and Y. Let X xor Y denote the bit-wise XOR of X and Y, and let XY denote the bit-wise AND of X and Y.

## 2. MD4 Algorithm Description

We begin by supposing that we have a b-bit message as input, and that we wish to find its message digest. Here b is an arbitrary nonnegative integer; b may be zero, it need not be a multiple of eight, and it may be arbitrarily large. We imagine the bits of the message written down as follows:

**m_0 m_1 ... m_{b-1}**

The following five steps are performed to compute the message digest of the message.

### Step 1. Append Padding Bits

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

**Step 2. Append Length**

A 64-bit representation of b (the length of the message before the padding bits were added) is appended to the result of the previous step. In the unlikely event that b is greater than 2^64, then only the low-order 64 bits of b are used. (These bits are appended as two 32-bit words and appended low-order word first in accordance with the previous conventions.)At this point the resulting message (after padding with bits and with b has a length that is an exact multiple of 512 bits. Equivalently, this message has a length that is an exact multiple of 16 (32-bit) words. Let M [0 ... N-1] denote the words of the resulting message, where N is a multiple of 16.

**Step 3. Initialize MD Buffer**

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

**word A: 01 23 45 67**
**word B: 89 ab cd ef**
**word C: fe dc ba 98**
**word D: 76 54 32 10**

**Step 4. Process Message in 16-Word Blocks**

We first define three auxiliary functions that each take as input three 32-bit words and produce as output one 32-bit word.

**F(X,Y,Z) = XY v not(X) Z**
**G(X,Y,Z) = XY v XZ v YZ**
**H(X,Y,Z) = X xor Y xor Z**

In each bit position F acts as a conditional: if X then Y else Z. The function F could have been defined using + instead of v since XY and not(X)Z will never have "1" bits in the same bit position.) In each bit position G acts as a majority function: if at least two of X, Y, Z are on, then G has a "1" bit in that bit position, else G has a "0" bit. It is interesting to note that if the bits of X, Y, and Z are independent and unbiased, the each bit of f(X,Y,Z) will be independent and unbiased, and similarly each bit of g(X,Y,Z) will be independent and unbiased. The function H is the bit-wise XOR or parity" function; it has properties similar to those of F and G.

Check out these exercises.

```
  /*   Process each 16-word block. */
     For i = 0 to N/16-1 do
       /* Copy block i into X. */
       For j = 0 to 15 do
         Set X[j] to M[i*16+j].
       end /* of loop on j */

       /* Save A as AA, B as BB, C as CC, and D as DD. */
       AA = A
       BB = B
       CC = C
       DD = D
       /* Round 1. */
       /* Let [abcd k s] denote the operation
            a = (a + F(b,c,d) + X[k]) <<< s. */
       /* Do the following 16 operations. */
       [ABCD  0  3]  [DABC  1  7]  [CDAB  2 11]  [BCDA  3 19]
       [ABCD  4  3]  [DABC  5  7]  [CDAB  6 11]  [BCDA  7 19]
       [ABCD  8  3]  [DABC  9  7]  [CDAB 10 11]  [BCDA 11 19]
       [ABCD 12  3]  [DABC 13  7]  [CDAB 14 11]  [BCDA 15 19]
       /* Round 2. */
       /* Let [abcd k s] denote the operation
            a = (a + G(b,c,d) + X[k] + 5A827999) <<< s. */
       /* Do the following 16 operations. */
       [ABCD  0  3]  [DABC  4  5]  [CDAB  8  9]  [BCDA 12 13]
       [ABCD  1  3]  [DABC  5  5]  [CDAB  9  9]  [BCDA 13 13]
       [ABCD  2  3]  [DABC  6  5]  [CDAB 10  9]  [BCDA 14 13]
       [ABCD  3  3]  [DABC  7  5]  [CDAB 11  9]  [BCDA 15 13]
       /* Round 3. */
       /* Let [abcd k s] denote the operation
            a = (a + H(b,c,d) + X[k] + 6ED9EBA1) <<< s. */
       /* Do the following 16 operations. */
       [ABCD  0  3]  [DABC  8  9]  [CDAB  4 11]  [BCDA 12 15]
       [ABCD  2  3]  [DABC 10  9]  [CDAB  6 11]  [BCDA 14 15]
       [ABCD  1  3]  [DABC  9  9]  [CDAB  5 11]  [BCDA 13 15]
       [ABCD  3  3]  [DABC 11  9]  [CDAB  7 11]  [BCDA 15 15]
       /* Then perform the following additions. (That is,
increment each of the four registers by the value it had before
this block was started.) */
       A = A + AA
       B = B + BB
       C = C + CC
       D = D + DD
     end /* of loop on i */
```

**Note:**

The value 5A...99 is a hexadecimal 32-bit constant, written with the high-order digit first. This constant represents the square root of 2. The octal value of this constant is **013240474631.** The value 6E..A1 is a hexadecimal 32-bit constant, written with the high-order digit first. This constant represents the square root of 3. The octal value of this constant is **015666365641.**

**Step 5. Output**

The message digest produced as output is A, B, C, and D. That is, we begin with the low-order byte of A, and end with the high-order byte of D. This completes the description of MD4.

**Summary:**

The MD4 message-digest algorithm is simple to implement, and provides a "fingerprint" or message digest of a message of arbitrary length. It is conjectured that the difficulty of coming up with two messages having the same message digest is on the order of 2^64 operations, and that the difficulty of coming up with any message having a given message digest is on the order of 2^128 operations. The MD4 algorithm has been carefully scrutinized for weaknesses. It is, however, a relatively new algorithm and further security analysis is of course justified, as is the case with any new proposal of this sort.

**3.3. MD5 algorithm:**

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

**1. MD5 Algorithm Description:**

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

**Step 1. Append Padding Bits:**

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

**Step 2. Append Length:**

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

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

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

**Step 4. Initialize MD Buffer/Chaining Variables**

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

**A: 01 23 45 67**
**B: 89 ab cd ef**
**C: fe dc ba 98**
**D: 76 54 32 10**

**Step 5. Process Message in 16-Word Blocks:**

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

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

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

**The inputs to each round are:**

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

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

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

**The process of rounds:**

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

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

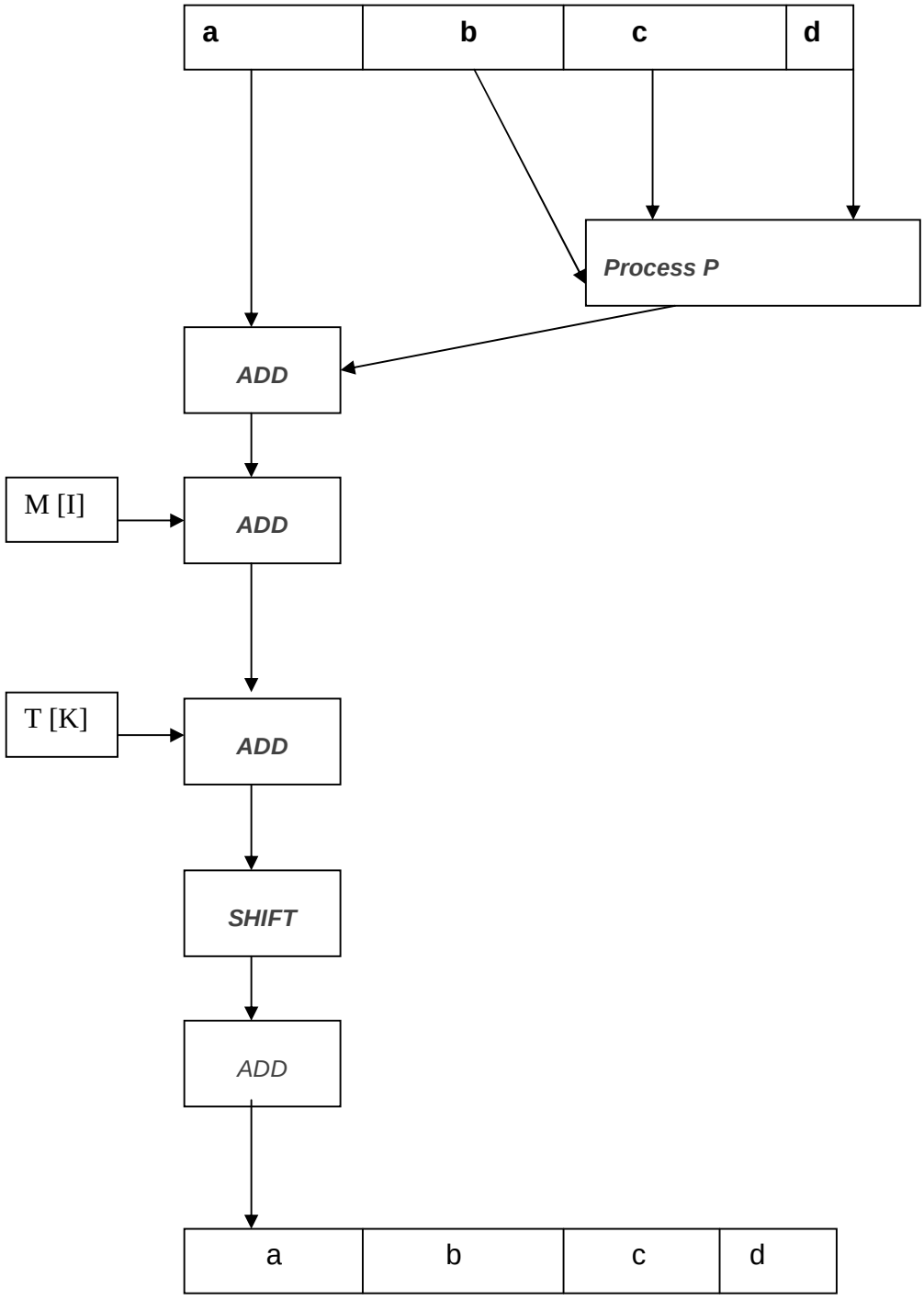
The message sub-block M[I] is added to the output of step 2.

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

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

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

**One MD5 Operation:**




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

Round 1 =(b and c) or (not (b)) and d

Round 2 = (b and d) or (c and (not(c)))

Round 3 = b xor c xor d

Round 4 = c xor (b or not (d))

**Summary:**

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

**Message Authentication Code (MAC)**

The concept of 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 involves in the case of message digests. In contrast, a MCA requires that the sender and the receiver should know a shared symmetric 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 receiver B. 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.

1. A then sends the original message M and the MAC h1 to B.

2. When b receives the message B also uses K to calculate its own MAC h2 over M.

3. B now compares h1 with h2. If the two matches, B concludes that the message M has not been changed during transit However, if h1 not equals h2, B rejects the message realizing that the message was changed during transit.

**The significance of a MAC is as follows:**

1. The MAC assures the receiver (in this case, B) that the message is not altered. This is because if an attacker alters the message but doesn't not alter the MAC (in this case, h1), then the receiver's calculation of the MAC (in this case h2) will differ from it. Why does the attacker then not also alter the MAC? Well, as we know, the key used in the calculation of the MAC (in this case K) is assumed to be known only to the sender and the receiver ( in this case, A and B). Therefore, the attacker does not know the key K, and therefore cannot alter the MAC.

2. The receiver (in this Case, B) is assured that the message indeed came from the correct sender (in this case, A). Since only the sender and the receiver (A and B, respectively, in this case) know the secret key ( in this case, K), no one else could have calculated the MAC( in this case, h1) sent by the sender ( in this case, A).

Interestingly, although the calculation of the MAC seems to be quite similar to an encryption process, it is actually different in one important respect. As we know, in symmetric key cryptography, the cryptographic process must be irreversible. That is, the encryption and the decryption are the mirror images of each other. However, note that in the case of MAC, both the sender and the receiver are performing encryption process only.

We have already discussed two main message digest algorithms, namely MD5 and SHA-

1. Can we reuse these algorithms for calculating a MAC, in their original form? Unfortunately, we cannot reuse them, because they do not involve the usage of a secret key, which is the basis of MAC. Consequently, we must have a separate practical algorithm implementation for the MAC. The solution is HMAC, a practical algorithm to implement MAC.

**SHS Padding stage or HMAC algorithm:**

**Introduction:**

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

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

**How does HMAC work?**

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 (eg. MD5, SHA-1, etc)

M = The input message whose MAC is to be calculated

L = The number of blocks in the message

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

Armed with these inputs, we shall use a step- by-step approach to understand the HMAC operation.

**STEP 1: Make the length of K equal to b**

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

- **Length of K < b**

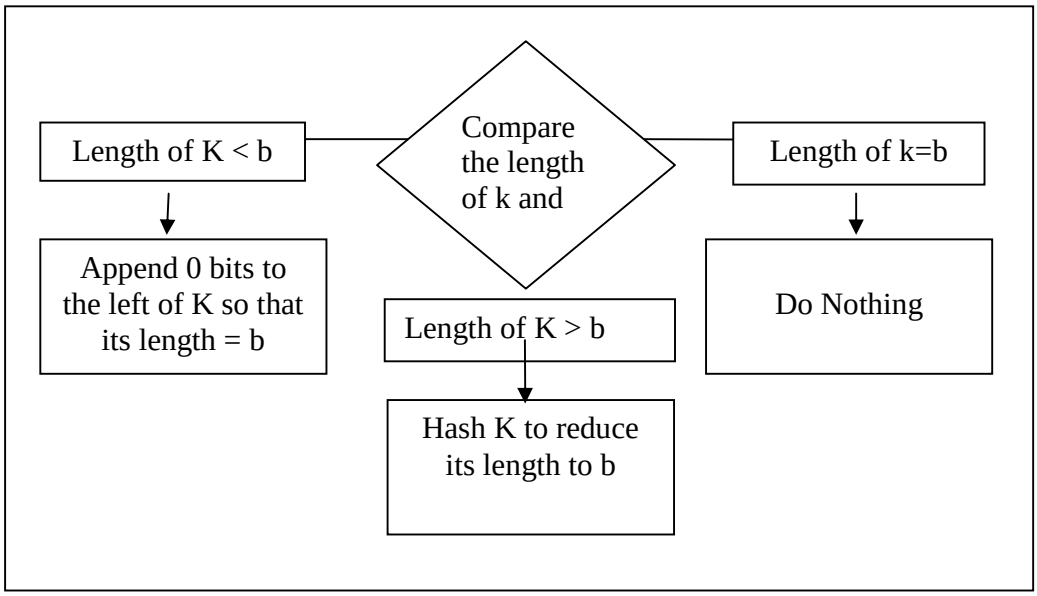
In this case, we need to expand the key K to make the length of K equal to the number of bits In the original message block (i.e. b). For this, we add as many as 0 bits as required to the left of K, For example, if the initial length of K = 170bits, and b = 512, then add 342 bits, all with a value 0,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

- **Length of K = b**

In this case, we do not take any action, and proceed to step2.

- **Length of K > b**

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




**Step 1 of HMAC**

**Step 2: XOR K with ipad to produce S1**

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

**Step 3: Append M to S1**

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

**Step 4: Message digest algorithm**

Now the selected message digest algorithm (e.g. MD5, SHA-1, etc.) is applied to the output of Step 3 (i.e. to the combination of S1 and M). Let us call the output of this operation as H. This is shown in Fig. 4.37.

**Step 5 : XOR K with opad to produce S2**

Now, we XOR K (the output of Step1) with opad to produce a variable called S2, This is shown in Fig. 4.38.

**Step 6: Append H to S2**

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

**Step 7: Message digest algorithm**

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

**Disadvantages of HMAC**

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

1. We assume in HMAC that the sender and the receiver only know about the symmetric key. However, we have studied in great detail that the problem of symmetric key exchange is quite serious, and cannot be solved easily. The same problem of key exchange is presently the case of HMAC.

2. Even if we assume that somehow the key exchange problem is resolved. HMAC cannot be used if the number of receivers is greater than one. This is because, to produce a MAC by using HMAC, we need to make use of a symmetric key. The symmetric key is supposed to be shared only by two parties: one sender and one receiver. Of course, this problem can be solved if multiple parties (one sender and all the receivers) share the same symmetric key. However, this resolution leads to a third problem.

3. The new problem is that how does a receiver know that the message was prepared and send by the sender, and not by the other receivers? After all, all the co-receivers also know the symmetric key. Therefore, it is quite possible that one of the co-receivers might have created a false message on behalf of the alleged sender, and using HMAC; the co-receiver might have prepared a MAC for the message, and sent the message and the MAC as if it originated at the alleged sender. There is no way to prevent or detect this.

4. Even if we somehow solve the above problem, one major concern remains. Let us go back to our simple case of one sender and one receiver. Now, only two parties – the sender (say A, a bank customer) and the receiver (say B, a bank) share the symmetric key secret. Suppose that one fine day, B transfers al the balance standing in the account of A to a third person's account, and closes A's bank account. A is shocked, and files a suit against B. In the court of law, B argues that A had sent an electronic message in order t perform this transaction, and produces that message as evidence. A claims that she never sent that message, and that it is a forged message. Fortunately, message produced by B as evidence also contained MAC, which was produced on the original message. As we know, only encrypting the message digests of the original message with the symmetric key shared by A and B could have produced it- and here is where the trouble is. Even though we have a MAC, how in the world are we now going to prove that the MAC was produced by A or by B? After all, both know about the shared secret key. It is equally possible for either of them to have created the original message and the MAC.

As it turns out, even if we are able to somehow resolve the first three problems, we have no solution for the fourth problem. Therefore, we cannot trust HMAC to be used in digital signatures.

**Questions for practice:**

1) What is message digest? Explain the idea behind digest.
2) List the pre-requisites for message digest.
3) Explain the concept of hash and prove how hash can be used to check the integrity.
4) Compare SHA-1 and MD5.
5) State and explain MD2 algorithm.
6) State and explain MD4 algorithm.
7) State and explain MD5 algorithm.
8) Explain about the SHS Padding stage or HMAC algorithm in detail.
9) State and explain the problems of HMAC.
10) What are the issues need to be addressed by HMAC? Also explain how it solves those issues.
11) Compare Message digest and HMAC.
12) State the advantages of HMAC.

# 4

# PUBLIC KEY CRYPTOGRAPHY

**In this unit, we are going to learn about the following topics:**

- Introduction
- Modular arithmetic
  - ✓ Meaning
  - ✓ Addition
  - ✓ Multiplication
  - ✓ Inverse Exponentiation
- RSA:
  - ✓ Generating keys
  - ✓ Encryption
  - ✓ Decryption.
- Other Algorithms
  - ✓ PKCS
  - ✓ Diffie-Hellman.
  - ✓ El-Gamal signatures
  - ✓ DSS
  - ✓ Zero-knowledge signatures

## 4.1 Introduction:

**The Public-Key Cryptography** Standards are specifications produced by RSA Laboratories in cooperation with secure systems developers worldwide for the purpose of accelerating the deployment of public-key cryptography. First published in 1991 as a result of meetings with a small group of early adopters of public-key technology, the PKCS documents have become widely referenced and implemented. Contributions from the PKCS series have become part of many formal and de facto standards, including ANSI X9 documents, PKIX, SET, S/MIME, and SSL.

## 4.2 Modular Arithmetic

**Meaning:**

Modular (often also Modulo) Arithmetic is an unusually versatile tool discovered by K.F.Gauss (1777-1855) in 1801. Two numbers a and b are said to be equal or congruent modulo N iff

N|(a-b), i.e. iff their difference is exactly divisible by N. Usually a,b, are nonnegative and N a positive integer. We write a = b (mod N).

When we write *n* mod *k* we mean simply the remainder when *n* is divided by *k*. Thus

**25 = 1 mod 3,**

**15 = 3 mod 4,**

**−13 = −3 mod 5**

**12 = 2 mod 5.**

It is an important fact, which again is most clearly seen using the theory of cosets that modular arithmetic respects sums and products. That is,

**(a+b) mod n = (a mod n)+ (b mod n) and**
**(a·b) mod n = (a mod n)·(b mod n)**

The set of numbers congruent to a modulo N is denoted $[a]_N$. If $b \in [a]_N$ then, by definition, N|(a-b) or, in other words, a and b have the same remainder of division by N. Since there are exactly N possible remainders of division by N, there are exactly N different sets $[a]_N$. Quite often these N sets are simply identified with the corresponding remainders: $[0]_N = 0$, $[1]_N = 1$,..., $[N-1]_N = N-1$. Remainders are often called **residues**; accordingly, [a]'s are also known as the **residue classes**.

It's easy to see that if a = b (mod N) and c = d (mod N) then (a+c) = (b+d) (mod N). The same is true for multiplication. These allows us to introduce an algebraic structure into the set $\{[a]_N: a=0,1,...,N-1\}$:

By definition,

1. $[a]_N + [b]_N = [a + b]_N$
2. $[a]_N \times [b]_N = [a \times b]_N$

Subtraction is defined in an analogous manner

$[a]_N - [b]_N = [a - b]_N$

And it can be verified that thus equipped set $\{[a]_N: a=0,1,...,N-1\}$ becomes a ring with commutative addition and multiplication. Division can't be always defined. To give an obvious example,

**$[5]_{10} \times [1]_{10} = [5]_{10} \times [3]_{10} = [5]_{10} \times [5]_{10} = [5]_{10} \times [7]_{10} = [5]_{10} \times [9]_{10} = [5]_{10}$.**
**$[5]_{10} \times [2]_{10} = [5]_{10} \times [4]_{10} = [5]_{10} \times [6]_{10} = [5]_{10} \times [8]_{10} = [5]_{10} \times [0]_{10} = [0]_{10}$.**

## An Introductory Example

Everyone knows that set of integers can be broken up into the following two classes:

- The even numbers (..., –6, –4, –2, 0, 2, 4, 6,...); and
- The odd numbers (..., –5, –3, –1, 1, 3, 5,…).

There are certain generalizations we can make about the arithmetic of numbers based on which of these two classes they come from. For example, we know that the sum of two even numbers is even. The sum of an even number and an odd number is odd. The sum of two odd numbers is even. The product of two even numbers is even, etc.

Modular arithmetic lets us state these results quite precisely, and it also provides a convenient language for similar but slightly more complex statements. In the above example, our **modulus i**s the number 2. The modulus can be thought of as the number of classes that we have broken the integers up into. It is also the difference between any two "consecutive" numbers in a given class.

Now we represent each of our two classes by a single symbol. We let the symbol "0" mean "the class of all even numbers" and the symbol "1" mean "the class of all odd numbers". There is no great reason why we have chosen the symbols 0 and 1; we could have chosen 2 and 1, or –32 and 177, but 0 and 1 are the conventional choices.

The statement "the sum of two even numbers is even" can be expressed by the following:

$$\mathbf{0 + 0 \equiv 0 \bmod 2.}$$

Here, the **"≡" symbol is not equality but congruence,** and the "mod 2" just signifies that our modulus is 2. The above statement is read "**Zero plus zero is congruent to zero, modulo two."** The statement "the sum of an even number and an odd number is odd" is represented by

$$\mathbf{0 + 1 \equiv 1 \bmod 2.}$$

Those examples are natural enough. But how do we write "the sum of two odd numbers is even"? It is the (at first strange looking) expression

$$\mathbf{1 + 1 \equiv 0 \bmod 2.}$$

Here the symbols "≡" and "mod 2" are suddenly very important! We have analogous statements for multiplication:

$$0 \times 0 \equiv 0 \bmod 2,$$
$$0 \times 1 \equiv 0 \bmod 2,$$
$$1 \times 1 \equiv 1 \bmod 2.$$

In a sense, we have created a number system with addition and multiplication but in which the only numbers that exist are 0 and 1. You may ask what use this has. Well, our number system is the system of **integers modulo 2,** and because of the previous six properties, **any arithmetic done in the integers translates to arithmetic done in the integers modulo 2**. This means that if we take any equality involving addition and multiplication of integers, say

$$12 \times 43 + 65 \times 78 = 5586,$$

Then reducing each integer **modulo 2** (i.e. replacing each integer by its class "representative" 0 or 1), then we will obtain a valid congruence. The above example reduces to

$$0 \times 1 + 1 \times 0 \equiv 0 \bmod 2, \text{ or } 0 + 0 \equiv 0 \bmod 2.$$

More useful applications of reduction modulo 2 are found in solving equations. Suppose we want to know which integers might solve the equation $3a - 3 = 12$.

Of course, we could solve for $a$, but if we didn't need to know what a is exactly and only cared about, say, whether it was even or odd, we could do the following. Reducing modulo 2 gives the congruence

$$1a + 1 \equiv 0 \bmod 2, \text{ or } a \equiv -1 \equiv 1 \bmod 2,$$

so any integer $a$ satisfying the equation $3a - 3 = 12$ must be odd. Since any integer solution of an equation reduces to a solution modulo 2, it follows that **if there is no solution modulo 2, and then there is no solution in integers**. For example, assume that $a$ is an integer solution to $2a - 3 = 12$, which reduces to $0 \cdot a + 1 \equiv 0 \bmod 2$,or $1 \equiv 0 \bmod 2$. This is a contradiction because 0 and 1 are different numbers modulo 2 (no even number is an odd number, and vice versa). Therefore the above congruence has no solution, so $a$ couldn't have been an integer. This proves that the equation $2a - 3 = 12$ has no integer solution.

Less trivially, consider the system of equations

$$\mathbf{6}\boldsymbol{a} \mathbf{- 5}\boldsymbol{b} \mathbf{= 4,}$$
$$\mathbf{2}\boldsymbol{a} \mathbf{+ 3}\boldsymbol{b} \mathbf{= 3.}$$

Modulo 2, these equations reduce to

$$\mathbf{0 + 1}\boldsymbol{b} \mathbf{\equiv 0 \bmod 2,}$$
$$\mathbf{0 + 1}\boldsymbol{b} \mathbf{\equiv 1 \bmod 2.}$$

This says that **b** is both even and odd, which is a contradiction. Therefore we know that the original system of equations has no integer solutions, and to prove this we didn't even need to know anything about **a**.

As shown by the preceding examples, one of the powers of modular arithmetic is the ability to show, often very simply, that certain equations and systems of equations have no integer solutions. Without modular arithmetic, we would have to find all of the solutions and then see if any turned out to be integers.

Of course, there is nothing special about the number 2. Any integer (except 0) will work for the modulus *m*. We now give the mathematical definition of congruence. Let $m \neq 0$ be an integer. We say that two integers **a and b** are **congruent modulo m** if there is an integer *k* such that $a - b = km$, and in this case we write $\mathbf{a \equiv b \bmod m.}$ Notice that the condition "$\boldsymbol{a - b = km}$ **for some integer** $\boldsymbol{k}$**"** is equivalent to the condition **"m divides a – b".**

In the previous section we used the modulus $m = 2$. Although we wrote congruences using only 0 and 1, really any integers are valid. Modulo 2, all of the even numbers are congruent to each other since the difference of any two even numbers is divisible by 2:

$$\mathbf{\ldots \equiv -6 \equiv -4 \equiv -2 \equiv 0 \equiv 2 \equiv 4 \equiv 6 \equiv \ldots \mod 2.}$$

Also, every odd number is congruent to every other odd number modulo 2 since the difference of any two odd numbers is even:

$$\mathbf{\ldots \equiv -5 \equiv -3 \equiv -1 \equiv 1 \equiv 3 \equiv 5 \equiv \ldots \mod 2.}$$

Therefore, anywhere we wrote "0" we could have written any other even number, and similarly "1" could have been replaced by any odd number. For example, instead of writing $0 \times 1 + 1 \times 0 \equiv 0 \bmod 2$, an equally valid statement would have been

$$\mathbf{10 \times (-13) + 27 \times 6 \equiv -122 \mod 2.}$$

Let's now look at other values for the modulus *m*. For example, let *m* = 3. All multiples of 3 are congruent to each other modulo 3 since the difference of any two is divisible by 3. Similarly, all numbers of the form 3*n* + 1 are congruent to each other, and all numbers of the form 3*n* + 2 are congruent to each other.

**... ≡ –9 ≡ –6 ≡ –3 ≡ 0 ≡ 3 ≡ 6 ≡ 9 ≡ ... mod 3.**
**... ≡ –8 ≡ –5 ≡ –2 ≡ 1 ≡ 4 ≡ 7 ≡ ... mod 3.**
**... ≡ –7 ≡ –4 ≡ –1 ≡ 2 ≡ 5 ≡ 8 ≡ ... mod 3.**

How about when *m* = 1? The difference of *any* two integers is divisible by 1, so all integers are congruent to each other modulo 1:

**... ≡ –3 ≡ –2 ≡ –1 ≡ 0 ≡ 1 ≡ 2 ≡ 3 ≡ ... mod 1.**

For this reason, *m* = 1 is not very interesting, and reducing an equation modulo 1 doesn't give any information about its solutions. The modulus *m* = 12 comes up quite frequently in everyday life, and its application illustrates a good way to think about modular arithmetic — the "clock arithmetic" analogy. If it's 7:00, what time will it be in 25 hours? Since 25 ≡ 1 mod 12, we simply add 1 to 7:

**7 + 25 ≡ 7 + 1 ≡ 8 mod 12.**

So the clock **will read 8:00**. Of course, we don't need the formality of modular arithmetic in order to compute this, but when we do this kind of computation in our heads, this is really what we are doing.

With **m** = 12, there are only 12 numbers ("hours") we ever need to think about. We count them 1, 2, 3... 10, 11, 12, 1, 2... starting over after 12. The numbers 1, 2, ..., 12 represent the twelve equivalence classes modulo 12: Every integer is congruent to exactly one of the numbers 1, 2, ..., 12, just as the hour on the clock always reads exactly one of 1, 2, ..., 12. These classes are given by

***12n + 1, 12n + 2, 12n + 3... 12n + 11, 12n*** as n ranges over the integers.

Of course, the minutes and seconds on a clock are also modular. In these cases the modulus is *m* = 60. If we think of the days of the week as labeled by the numbers 0, 1, 2, 3, 4, 5, 6, then the modulus is *m* = 7. The point is that we measure many things, both in mathematics and in real life, in periodicity, and this can usually be thought of as an application of modular arithmetic.

Most of the cryptosystems uses modular arithmetic for its encryption and decryption. For the key distribution problems and algorithms, they used modular arithmetic. That is why it is important for us to know about the fundamentals about modular arithmetic. Let us see the algorithms that support modular arithmetic.

### 4.2.1. Why asymmetric key cryptography?

**In a real practical life, key distribution is the biggest problem in symmetric key cryptography.**

**Problem of Key distribution in Symmetric Key cryptography:**

As in case of symmetric key cryptography, the key that has to be used for both encryption and decryption should be the 'same' this leads to a problem that how the two parties requiring secure communication can 'agree' or 'decide' upon a common key, without letting any third person know about it? There can be many ways in which the two parties will try to communicate assuming it is secure, but it may not be so. e.g. even if they exchange letters, seal envelopes into locked boxes, talk over open media for the common key, or send the key along with the locked boxes, whatever may be the means used, it turns out to be practically non-viable or difficult to implement.

That is to say, there are very much chances of intercepting the communication between two parties if any of these methods are used. This is called the 'problem of key distribution'.

In order to come out of this problem, one good solution was given by two scientists jointly known as 'Diffie-Hellman key exchange algorithm'.

**1. The Diffie-Hellman Key exchange algorithm:**

Whitefield Diffie and Martin Hellman, in 1976 have come out with a good solution to the problem of key distribution as mentioned above. The steps of this algorithm are given in Unit-6. (It must be noted, that this is NOT an encryption or decryption algorithm but is only used for agreeing upon a. symmetric key. Once it is done, some specific algorithm should be used for the purpose of encryption/decryption.) Let us analyze the problems with the algorithm now.

**2. Problems with the algorithm:**

Although, it is seen that this algorithm turns out to be a good solution to the above mentioned key distribution problem, still it does not solve all the problems! This is because the algorithm can

fail if a hacker makes what is called as the **man-in-the-middle attack.** This way, even though the two parties will feel that they are talking to each other, practically they are in-turn communicating with the hacker as he places himself in between them and switch back and forth the communication.

**For example,**

1. Alice wants to communicate with Bob securely. For this purpose, she sends the values of n and g to Bob. Let n=11 and g=7.
2. Alice does not realize that the attacker Tomis listening quitely ;to the conversation between her and Bob.Tom simply picks up the values of n and g, and also forwards them to Bob as they originally were.

| Alice | Tom | Bob |
|---|---|---|
| N=11,g=7 | N=11,g=7 | N=11,g=7 |

3. Now, let us assume that Alice, Tom and Bob select random numbers x and y.

| Alice | Tom | Bob |
|---|---|---|
| X=3 | x=8,y=6 | y=9 |

4. Alice calculates A and Bob calculates B whereas Tom calculates both A and B to play the role of man in middle.

| Alice | Tom | Bob |
|---|---|---|
| $A=g^x \bmod n$ | $A=g^x \bmod n$ | $B= g^y \bmod n$ |
| $=7^3 \bmod 11$ | $=7^8 \bmod 11$ | $=79 \bmod 11$ |
| $=343 \bmod 11$ | $= 5764801 \bmod 11$ | $= 40353607 \bmod 11$ |
| $=2$ | $= 9$ | $=8$ |
| | $B = g^y \bmod n$ | |
| | $= 7^6 \bmod 11$ | |
| | $= 117649 \bmod 11$ | |
| | $= 4$ | |

5. Alice send her A2 to Bob. Tom intercepts it and send his A 9 to him.
   - In return , Bob sends his B 8 to Alice. Tom intercepts it and send his B 4 to Alice.
   - Based on these values, all the three persons now calculate their keys.

| Alice | Tom | Bob |
|---|---|---|
| K1=Bx mod n<br>=43 mod 11<br>=64 mod 11<br>**=9** | K1=Bx mod n<br>=88 mod 11<br>=16777216mod 11<br>=5<br>k2=Ay mod n<br>=26 mod 11<br>=64 mod 11<br>**=9** | K2=Ay mod n<br>=99 mod 11<br>=387420489 mod 11<br>=5 |

As we can see, **the MITM** attack can work against the Diffie-Hellman Key exchange algorithm, causing it to fail. This is plainly because the person in middle makes the actual communicators believe that they are talking to each other, whereas they are actually talking to he man-in-the middle, who is talking to each of them.

The second problem is regarding **the no. Of keys required.** In our example, we have just seen the situation of only two communicating parties. What would be the situation if a third party say 'third' is added!

One must think of the situation when communication between first-second, second-third as well as third-first must be secure! This would obviously require three keys! Then assume how many keys would be required to securely communicate between 1000 people that to independently?

To find out this answer, one formula is used. It says, the total no. of keys required to securely communicate between 'n' individuals is = n (n-l) / 2. Hence in our example for 1000 people, 1000(999)/2 = 499500 keys would be needed. This certainly increases the complications further.n order to recover from these problems, the second technique (mentioned in the beginning) comes into picture, i.e. the Asymmetric Key cryptography. This states that two types of keys would be required, one each for encryption and decryption.

### 4.2.2. Generating Keys:

**1. The concept of Public key and Private key:**

The Asymmetric key cryptography is also known as a 'public key cryptography', which uses a key-pair rather than a single key. The importance of this scheme is that only one key-pair is required to securely communicate between any numbers of other parties. ( unlike the huge no. of keys that we've seen with earlier method.) Hence, one problem is overcome right away. One of these two keys is called public key (which can be announced to the world) and another is private key (obviously to be kept with oneself). This is to be followed by everyone who wants to communicate securely.

**2 .The working of public and private keys:**

Asymmetric key cryptography (using public and private keys) works as under:

Consider the scenario, X wants to send a message to Y, without having to worry about its security.

1. Then X and Y should each have a private key and a public key.
   - **X should keep its private key secret.**
   - **Y should keep its private key secret**.
   - X should inform Y about its public key.
   - Y should inform X about its public key

   (Both now have their own set of keys ready. )

2. When X wants to send message to Y, X encrypts with Y's public key (as it is known to everyone)
3. X then sends this message to Y.
4. Then, Y decrypts this message using his own private key (known only to Y)

[This ensures in this case, that the message can be encrypted & sent by anyone, but can only be decrypted by Y. Hence, any interception will not result in knowing the sensitive information as key is only with Y.]

Similarly, on the other side, if Y wants to send the message to X, reverse method is performed.

5. Y encrypts the message using X's public key and sends this to X
6. On receiving the message, X can further decrypt it using his own private key.

The basis of this working lies in the assumption of large prime number with only two factors. If one of the factors is used for encryption process, only the other factor shall be used for decryption.

The **best example** of an asymmetric key cryptography algorithm is the famous **RSA algorithm** (developed by **Rivest, Shamir and Adleman** at MIT in 1978, based on the framework setup by Diffie & Hellman earlier).

In public-key encryption/decryption, the public key that is used for encryption is different from the private key that is used for decryption. The public key is available to the public; the private key is available only to an individual. **Public-key encryption/ decryption has two advantages.**

First, it removes the **restriction of a shared symmetric key** between two entities (e.g., persons) that need to communicate with each other. A shared symmetric key is shared by the two parties and cannot be used when one of them wants to communicate with a third party. In public-key encryption! Decryption, each entity creates a pair of keys; the private one is kept, and the public one is distributed. Each entity is independent, and the pair of keys created can be used to communicate with any other entity.

**The second advantage** is that the number of keys needed is reduced tremendously.

In this system, for I thousand users to communicate, only 1 thousand pairs of keys i.e. 2000 keys are needed, not 4,99,500, as was the case in symmetric-key cryptography.

**3.. Public-key cryptography also has two disadvantages.**

The big disadvantage is the **complexity of the algorithm**. If we want the method to be effective, the algorithm needs large numbers. Calculating the cipher text from plaintext using the long keys takes a lot of time. That is the main reason that public-key cryptography is not recommended for large amounts of text.

**Public-key algorithms are more efficient for short messages.**

The second disadvantage of the public-key method is that the association between an entity and its public key must be

verified. If Alice sends her public key via an email to Bob, then Bob must be sure that the public key really belongs to Alice and nobody else.

### 4.3. RSA Algorithm:

1. Generate two large random primes, *p* and *q*, of approximately equal size
2. Calculate N = PXQ
3. Select the public key that is the encryption key E such that it is not a factor of (p-1)(q-1).
4. Select the private key that is the decryption key D such that the following equation is true:

   **(DXE) mod (P-1) X (Q-1)=1**

5. For encryption, calculate the cipher text CT as $CT=PT^E$ mod N.
6. Send CT as the cipher text to the receiver.
7. For decryption, calculate the plain text PT as $PT=CT^D$ mod N.

### A very simple example of RSA encryption:

This is an extremely simple example using numbers you can work out on a pocket calculator (those of you over the age of 35 can probably even do it by hand).

1. Select primes p=11, q=3.
2. n = pq = 11.3 = 33
   phi = (p-1)(q-1) = 10.2 = 20
3. Choose e=3
   Check gcd(e, p-1) = gcd(3, 10) = 1 (i.e. 3 and 10 have no common factors except 1),
   and check gcd(e, q-1) = gcd(3, 2) = 1
   therefore gcd(e, phi) = gcd(e, (p-1)(q-1)) = gcd(3, 20) = 1
4. Compute d such that ed ≡ 1 (mod phi)
   i.e. compute $d = e^{-1}$ mod phi = $3^{-1}$ mod 20
   i.e. find a value for d such that phi divides (ed-1)
   i.e. find d such that 20 divides 3d-1.
   Simple testing (d = 1, 2, ...) gives d = 7
   Check: ed-1 = 3.7 - 1 = 20, which is divisible by phi.
5. Public key = (n, e) = (33, 3)
   Private key = (n, d) = (33, 7).

This is actually the smallest possible value for the modulus n for which the RSA algorithm works.

Now say we want to encrypt the message m = 7,
$c = m^e \bmod n = 7^3 \bmod 33 = 343 \bmod 33 = 13$.
Hence the ciphertext c = 13.

To check decryption we compute
$m' = c^d \bmod n = 13^7 \bmod 33 = 7$.
Note that we don't have to calculate the full value of 13 to the power 7 here. We can make use of the fact that
$a = bc \bmod n = (b \bmod n).(c \bmod n) \bmod n$
so we can break down a potentially large number into its components and combine the results of easier, smaller calculations to calculate the final value.

One-way of calculating m' is as follows: -
$m' = 13^7 \bmod 33 = 13^{(3+3+1)} \bmod 33 = 13^3.13^3.13 \bmod 33$
$= (13^3 \bmod 33).(13^3 \bmod 33).(13 \bmod 33) \bmod 33$
$= (2197 \bmod 33).(2197 \bmod 33).(13 \bmod 33) \bmod 33$
$= 19.19.13 \bmod 33 = 4693 \bmod 33$
$= 7$.

## 4.4. Other Algorithms

### 4.4.1. DIGITAL SIGNATURE STANDARD (DSS):

When a message is received, the recipient may desire to verify that the message has not been altered in transit. Furthermore, the recipient may wish to be certain of the originator's identity. Both of these services can be provided by the DSA. A digital signature is an electronic analogue of a written signature in that the digital signature can be used in proving to the recipient or a third party that the message was, in fact, signed by the originator. Digital signatures may also be generated for stored data and programs so that the integrity of the data and programs may be verified at any later time.

**Explanation:** This Standard specifies a Digital Signature Algorithm (DSA) appropriate for applications requiring a digital rather than written signature. The DSA digital signature is a pair of large numbers represented in a computer as strings of binary digits. The digital signature is computed using a set of rules (i.e., the DSA) and a set of parameters such that the identity of the signatory and integrity of the data can be verified. The DSA provides the capability to generate and verify signatures. Signature generation makes use of a private key to generate a digital signature.

Signature verification makes use of a public key which corresponds to, but is not the same as, the private key. Each user possesses a private and public key pair. Public keys are assumed to be known to the public in general. Private keys are never shared. Anyone can verify the signature of a user by employing that user's public key. Signature generation can be performed only by the possessor of the user's private key.

A hash function is used in the signature generation process to obtain a condensed version of data, called a message digest .The message digest is then input to the DSA to generate the digital signature. The digital signature is sent to the intended verifier along with the signed data (often called the message). The verifier of the message and signature verifies the signature by using the sender's public key. The same hash function must also be used in the verification process. Similar procedures may be used to generate and verify signatures for stored as well as transmitted data. The procedure is given in fig 4.1.

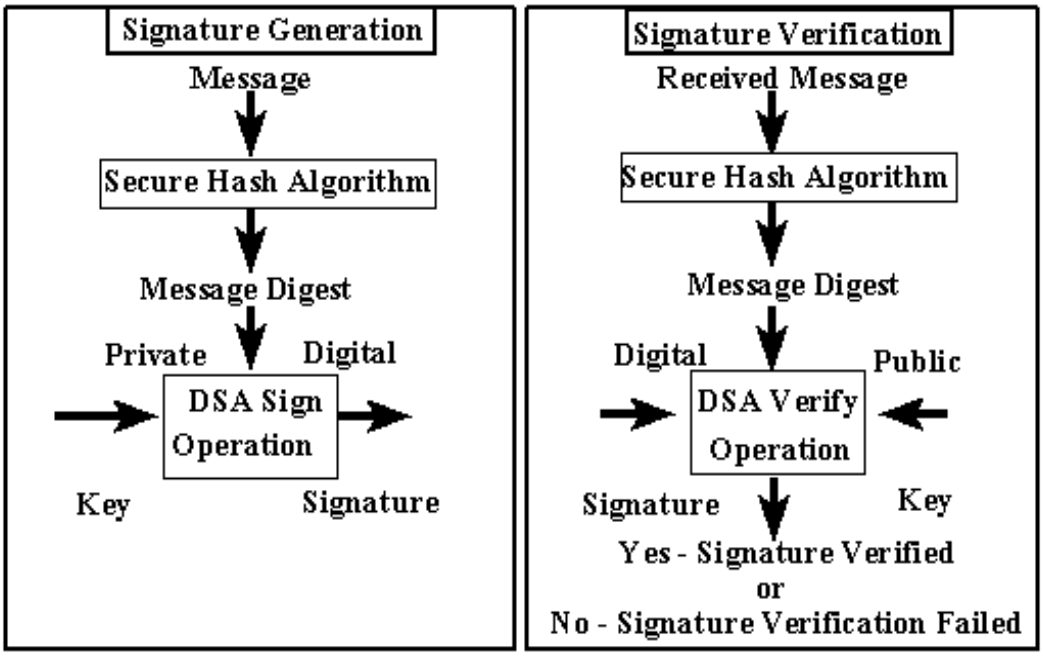



**Figure 4.1: Using the SHA with the DSA**

## 1. USE OF THE DSA ALGORITHM

The DSA is used by a signatory to generate a digital signature on data and by a verifier to verify the authenticity of the signature. Each signatory has a public and private key. The private key is used in the signature generation process and the public key is used in the signature verification process. For signature

generation and verification, the data which is referred to as a message, M is reduced by means of the Secure Hash Algorithm (SHA. An adversary, who does not know the private key of the signatory, cannot generate the correct signature of the signatory. In other words, signatures cannot be forged. However, by using the signatory's public key, anyone can verify a correctly signed message.

A means of associating public and private key pairs to the corresponding users is required. That is, there must be a binding of a user's identity and the user's public key. This binding may be certified by a mutually trusted party. For example, a certifying authority could sign credentials containing a user's public key and identity to form a certificate.

## 2. SIGNATURE GENERATION:

The signature of a message M is the pair of numbers r and s computed according to the equations below:

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

$s = (k^{-1}(SHA(M) + xr)) \bmod q.$

In the above, $k^{-1}$ is the multiplicative inverse of k, mod q; i.e., $(k^{-1} k) \bmod q = 1$ and $0 < k^{-1} < q$. The value of SHA (M) is a 160-bit string output by the Secure Hash Algorithm. For use in computing, this string must be converted to an integer. As an option, one may wish to check if r = 0 or s = 0. If either r = 0 or s = 0, a new value of k should be generated and the signature should be recalculated. The signature is transmitted along with the message to the verifier.

## 3. SIGNATURE VERIFICATION:

Prior to verifying the signature in a signed message, p, q and g plus the sender's public key and identity are made available to the verifier in an authenticated manner. Let M', r' and s' be the received versions of M, r, and s, respectively, and let y be the public key of the signatory. To verifier first checks to see that $0 < r' < q$ and $0 < s' < q$; if either condition is violated the signature shall be rejected. If these two conditions are satisfied, the verifier computes

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

$$\mathbf{u1 = ((SHA(M')w) \bmod q}$$

$$\mathbf{u2 = ((r')w) \bmod q}$$

$$\mathbf{v = (((g)^{ul} (y)^{u2}) \bmod p) \bmod q.}$$

If v = r', then the signature is verified and the verifier can have high confidence that the received message was sent by the

party holding the secret key x corresponding to y. If v does not equal r', then the message may have been modified, the message may have been incorrectly signed by the signatory, or the message may have been signed by an impostor. The message should be considered invalid.

**Applicability:** This standard is applicable to all Federal departments and agencies for the protection of unclassified information that is not subject to section 2315 of Title 10, United States Code, or section 3502(2) of Title 44, United States Code. This standard shall be used in designing and implementing public-key based signature systems which Federal departments and agencies operate or which are operated for them under contract. Adoption and use of this standard is available to private and commercial organizations.

## 4. RANDOM NUMBER GENERATION FOR THE DSA

Any implementation of the DSA requires the ability to generate random or pseudorandom integers. Such numbers are used to derive a user's private key, x, and a user's per message secret number, k. These randomly or pseudo randomly generated integers are selected to be between 0 and the 160- bit prime q (as specified in the standard).

**Applications:** The DSA authenticates the integrity of the signed data and the identity of the signatory. The DSA may also be used in proving to a third party that data was actually signed by the generator of the signature. The DSA is intended for use in electronic mail, electronic funds transfer, electronic data interchange, software distribution, data storage, and other applications which require data integrity assurance and data origin authentication.

**Implementations:** The DSA may be implemented in software, firmware, hardware, or any combination thereof. NIST (National Institute of Standards and Technology) is developing a validation program to test implementations for conformance to this standard.

**Qualifications:** The security of a digital signature system is dependent on maintaining the secrecy of users' private keys. Users must therefore guard against the unauthorized acquisition of their private keys. While it is the intent of this standard to specify general security requirements for generating digital signatures, conformance to this standard does not assure that a particular implementation is secure. The responsible authority in each agency or department shall assure that an overall implementation provides an acceptable level of security. This standard will be reviewed every five years in order to assess its adequacy.

**Waiver Procedure:** Under certain exceptional circumstances, the heads of Federal departments and agencies may approve waivers to Federal Information Processing Standards (FIPS). Waiver shall be granted only when:

a. Compliance with a standard would adversely affect the accomplishment of the mission of an operator of a Federal computer system; or

b. Compliance with a standard would cause a major adverse financial impact on the operator which is not offset by Government-wide savings.

Agency heads may act upon a written waiver request containing the information detailed above. Agency heads may also act without a written waiver request when they determine that conditions for meeting the standard cannot be met. Agency heads may approve waivers only by a written decision which explains the basis on which the agency head made with required finding(s).

## ElGamal Signature Scheme:

### Introduction.

Before discussing the ElGamal signature scheme, let's first talk about the ElGamal public key scheme. ElGamal is based on the problem of Discrete Logarithms in the group $(Z_p^*, \bullet)$. Let p be a prime and let $\alpha$ be a primitive element in $Z_p^*$. The set of all plaintexts $P = Z_p^*$, the set of all ciphertexts $C = Z_p^* \times Z_p^*$, and the set of all keys $K = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \bmod p\}$. p, $\alpha$, and $\beta$ make up the public key and the private key is p, $\alpha$, and a. Alice chooses a secret random number k in $Z_{p-1}$, and given m as the plaintext to be encrypted we have:

### Encryption

$e_k(m, k) = (y1, y2)$, where

$y_1 = \alpha^k \bmod p$ and

$y_2 = (m * \beta^k) \bmod p$

### Decryption

$m = d_k(y_1, y_2) = y_2(y_1^a)^{-1} \bmod p$, for all $y_1, y_2$ in $Z_p^*$ (notice the multiplicative inverse $(y_1^a)^{-1}$)

**Cryptosystem Example.**

Let p = 7, α = 2, a = 5, m = 6 and k = 4. Alice computes the following:

$\beta = 2^5 \bmod 7 = 4$
$y_1 = 2^4 \bmod 7 = 2$
$y_2 = [6(4^4)] \bmod 7 = 1536 \bmod 7 = 3$

Alice then sends (2, 3) to Bob, and Bob decrypts the ciphertext as:

$m = [3(2^5)^{-1}] \bmod 7$
$= [(3 \bmod 7)(32^{-1} \bmod 7)] \bmod 7$
$= [3 * 2] \bmod 7 = 6$

The ElGamal signature scheme is as follows:

Again let p be a prime and let α be a primitive element in $Z_p^*$. The set of all plaintexts $P = Z_{p-1}$, the set of all ciphertexts $C = Z_{p-1} \times Z_{p-1}$, and the set of all keys $K = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \bmod p\}$. p, α, and β make up the public key and the private key is p, α, and a. Alice chooses a secret random number k in $Z_{p-1}$, and given m as the plaintext to be signed we have:

**Signing**

$sig_k(m, k) = (y_1, y_2)$, where

$y_1 = \alpha^k \bmod p$ and

$y_2 = [(m - a * y_1)(k^{-1})] \bmod (p - 1)$

**Verification:**

$ver_k(m, (y_1, y_2)) \leftrightarrow y_1^{y2} * \beta^{y1} = \alpha^m \bmod p$

**Signature Example.**

Let p = 17, α = 2, a = 4, m = 6 and k = 3. Alice computes the following signature:

$\beta = 2^4 \bmod 17 = 16$
$y_1 = 2^3 \bmod 17 = 8$
$y_2 = [(6 - 4 * 8)(11)] \bmod 16 = -286 \bmod 16 = -14 \bmod 16 = 2$

Alice then sends (6, 8, 2) to Bob, and Bob verifies the signature as:

$2^6 \equiv [(8^2)(16^8)] \bmod 17 \rightarrow 2^6 \bmod 17 = [(8^2)(16^8)] \bmod 17$
$= 13$ It checks.

**Security:**

If the attacker can compute the value $a = \log_{\alpha}\beta$, then ElGamal signatures can be forged. As long as p is chosen carefully and α is a primitive element modulo p, then solving the Discrete Logarithm problem in $Z_p^*$ is infeasible. Additionally, k must be secret, only used once and random. As shown above, ElGamal can also be used for encryption as well, but the messages should be relatively small in size.

## 5. Zero-Knowledge Signatures

### ZK-SSH - A Zero Knowledge Implementation for OpenSSH

### 1. Motivation

SSH provides a secure way of accessing remote systems. This goal can only be achieved if it incorporates user authentication which ensures the identity of a user. Several methods like public key, password and host-based authentication methods are implemented in OpenSSH. Public key systems like RSA and DSA provide a reasonable level of security in regard to properties like transferability and impersonation. Nevertheless, these challenge response methods leak in polynomial time information to a third party making it easier to impersonate another party. Although this is nothing to be really afraid of, this problem does not exist when a zero knowledge user authentication protocol is used. Therefore, a zero knowledge protocol was chosen implemented for usage with OpenSSH.

### 2. Idea behind zero knowledge

Imagine that Alice knows a secret and wants to prove this to Bob. Of course she could just reveal the secret to Bob, but then he would know the secret too. Instead Alice does something like this:

Alice knows the secret password for Alibaba's cave. Alice goes into the cave and randomly chooses the left or right corridor. Then Bob enters the cave and tells Alice from which corridor she should walk out from. Let's say Bob picked the left corridor. If Alice chooses the left corridor at the beginning, she can just walk the same way back and does not even have to use the password. If she is standing on the right side, she has to use the password, the magic wall opens and she also can walk into the left corridor.

At this point Bob is not very impressed. Alice had a 50% chance of already being in the left corridor. So she could have passed the test, even without knowing the password. So Bob plays the same game another round. If Alice passes the test again, Bob is a little more convinced. After a sufficient amount of rounds Bob will believe that Alice knows the secret to Alibaba's cave. For example, after 10 rounds of playing this game, Alice has got a chance of (1/2) to the power of 10, or in numbers 0.0009765625% to pass the test if she does not know the secret. We shall use these facts in a decisive manner below.

Every process of encryption and decryption is necessarily associated with a **'key'**- the combination used for encryption and/or decryption, and an algorithm i.e. the rules or steps used for both encryption and decryption. The requirement of 'same' key as in case of 'symmetric' key cryptography leads to a common problem called 'problem of key distribution', i.e. how the two parties should agree upon a 'common' key that has to be used for the process. This is as described below.

### 4.6. PKCS

In cryptography, **PKCS** refers to a group of **public Key Cryptography Standards** devised and published by RSA Security. RSA Data Security Inc was assigned the licensing rights for the patent on the RSA asymmetric key algorithm and acquired the licensing rights to several other key patents as well.As such, RSA Security and its research division, RSA Labs, were interested in promoting and facilitating the use of public-key techniques. To that end, they developed the PKCS standards. They retained control over them, announcing that they would make changes/improvements as they deemed necessary, and so the PKCS standards were not, in a significant sense, actual industry standards, despite the name. Some, but not all, have in recent years begun to move into "standards track" processes with one or more of the relevant standards organizations.

The **P**ublic-**K**ey **C**ryptography **S**tandards are specifications produced by RSA Laboratories in cooperation with secure systems developers worldwide for the purpose of accelerating the

deployment of public-key cryptography. First published in 1991 as a result of meetings with a small group of early adopters of public-key technology, the PKCS documents have become widely referenced and implemented. Contributions from the PKCS series have become part of many formal and de facto standards, including ANSI X9 documents, PKIX, SET, S/MIME, and SSL.

| PKCS Standards Summary | | | |
|---|---|---|---|
| | **Version** | **Name** | **Comments** |
| **PKCS #1** | 2.1 | RSA Cryptography Standard | Defines the mathematical properties and format of RSA public and private keys, and the basic algorithms and encoding/padding schemes for performing RSA encryption, decryption, and producing and verifying signatures. |
| **PKCS #2** | - | **Withdrawn** | No longer active. Covered RSA encryption of message digests, but was merged into PKCS #1. |
| **PKCS #3** | 1.4 | Diffie-Hellman Key Agreement Standard | A cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. |
| **PKCS #4** | - | *Withdrawn* | No longer active. Covered RSA key syntax, but was merged into PKCS #1. |
| **PKCS #5** | 2.0 | Password-based Encryption Standard | Refer RFC 2898 and PBKDF2. |
| **CS #6** | 1.5 | Extended-Certificate Syntax Standard | Defines extensions to the old v1 X.509 certificate specification. Obsolete by v3 of the same. |
| **PKCS #7** | 1.5 | Cryptographic Message Syntax Standard | Used to sign and/or encrypt messages under a PKI. Used also for certificate dissemination. Formed the basis for S/MIME, which is as of 2009 based on RFC 3852, an updated Cryptographic Message Syntax Standard (CMS). Often used for single sign-on. |
| **PKCS #8** | 1.2 | Private-Key Information Syntax Standard. | Used to carry private certificate key pairs (encrypted or unencrypted). |

| | | | |
|---|---|---|---|
| **PKCS #9** | 2.0 | Selected Attribute Types | Defines selected attribute types for use in PKCS #6 extended certificates, PKCS #7 digitally signed messages, PKCS #8 private-key information, and PKCS #10 certificate-signing requests. |
| **PKCS #10** | 1.7 | Certification Request Standard | Format of messages sent to a certification authority to request certification of a public key. See certificate signing request. |
| **PKCS #11** | 2.20 | Cryptographic Token Interface (Cryptoki) | An API defining a generic interface to cryptographic tokens (see also Hardware Security Module). Often used in single sign-on, Public-key cryptography and disk encryption systems. |
| **PKCS #12** | 1.0 | Personal Information Exchange Syntax Standard | Defines a file format commonly used to store private keys with accompanying public key certificates, protected with a password-based symmetric key. PFX is a predecessor to PKCS#12.<br>This container format can contain multiple embedded objects, e.g. multiple certificates. Usually protected/encrypted with a password. Can be used as a format for the Java key store. Usable by Tomcat, but NOT by Apache. |
| **PKCS #13** | – | Cryptography Standard | (Under development.) |
| **PKCS #14** | – | Pseudo-random Number Generation | (Under development.) |
| **PKCS #15** | 1.1 | Cryptographic Token Information Format Standard | Defines a standard allowing users of cryptographic tokens to identify themselves to applications, independent of the application's Cryptoki implementation (PKCS #11) or other API. |

**(Source: www.wikipedia.org)**

**Questions:**

1. What is modular arithmetic? Explain it in detail. How it can be used in cryptography?
2. “RSA Algorithm is the best one in generating public keys and private keys as well as the robust one in cryptographic world”- Explain this statement.
3. Prove RSA Algorithm with example.
4. Explain “Elgamal Signature Scheme” in detail.
5. Explain how DSS ensures verification and validation.
6. State the properties and applications of DSS,
7. What is PKCS? State the standards of PKCS.
8. Discuss the problems with Diffie-Hellman algorithm.
9. Explain “Bucket-Brigade Attack or MITM attack” in detail.
10. What do you understand by “Zero knowledge signature”? Explain in detail.
11. How zero-knowledge proof will be efficient when it is integrated with SSH?

# 5

# AUTHENTICATION

1. Password based
    1. Distributed systems,
    2. On-line vs. off-line guessing,
    3. Storing.
2. Cryptographic authentication:
    1. Passwords as keys,
    2. Protocols
    3. KDC's,
3. Address based :
    1. Certification Revocation,
    2. Inter-domain, groups, delegation.
    3. Authentication of People
4. Verification techniques
    1. passwords, length of passwords, password distribution,
    2. Smart cards
    3. Biometrics

## 1. Password based authentication:

Passwords to the systems are something like the keys to the front doors. A front door is likely to be the first that is attacked by an intruder. Making use of login-name & password is an easy & cheap method of authentication, and is the most widely used. The passwords and the corresponding usernames are stored with the server database. Whenever the user enters into the system, the password entered by the user will be forwarded to the server and the verification with the database will be performed and the authentication will be completed. The storage of password with the server and the transmission password from the client to the server may create security breach. The various types of attacks possible with the password are discussed below.

### 1.1 On-line vs. off-line guessing:

Certain techniques are cracking the password directly by creating some hackers programme or by creating the fake login screen. This will attack the password directly either by using certain

permutation combination or by the support of the dictionary. The passwords can also be attacked offline by using social engineering techniques or by casual discussions with the user. The following are some methods which will attack the password either offline or online.

1. **Direct Approach:** This method apparently seems to be very easy but yet is very ineffective, since nobody is going to disclose the password easily to anyone. Still, intruders use this as the first method & try their luck & then go for more difficult ones. This attack is performed in such a way that the intruders will try with the various keywords, which are directly related with the user like name, relatives name, organization, city, designation etc. Normally user will select the passwords which can be easily remembered by them and which are directly related with them. The intruders will use this loophole and directly attack the password.

2. **Dictionary based attacks*:*** Hackers may use ready-made dictionaries for checking the passwords of systems, using special software. This method is somewhat difficult and time consuming, but not very sure. The main problem with this method is that the dictionary will be containing only the meaningful combination of characters, whereas the password can be of any combination of characters. The average of success will be very less with this method.

3. **Brute force attacks*:*** This involves using several combinations of keys such as alphabets, numbers, special characters etc. for a specific no. Of digits & comparing or applying them to guess passwords. Normally the possibility of getting the password is very easy with this method. Because of the possibility of the password is the only combination of the keys available within the keypad. By combining the same keys with various combinations the passwords can be easily cracked. This refers to the process of trial and error method with lots of permutations and combinations. Of course this is most tedious and time consuming, but surer method of getting passwords.

4. **Using fake / login*:*** Sometimes, if the attacker is an insider, or is someone who can get a direct access, then the attacker may keep a fake login program running on a terminal, which feels legitimate to unknown users. When someone logins, he gets an invalid login message, and the password is meanwhile collected somewhere, which is available for the attacker. Windows prevents from this attack, by requiring Ctrl-Alt-Del keys before login. Another way to protect is to always lock the terminal while going away from it.

5. **Packet sniffing*:*** As another attempt, attackers may intercept the packets flowing through the network. Some protocols let out the passwords in clear text while transmitting, which may be grabbed by attackers sniffing the packets. Although this is also tedious, it may work out sometimes. Packet sniffing is a form of wiretap applied to computer networks instead of phone networks. It came into vogue with Ethernet, which is known as a "shared medium" network. This means that traffic on a segment passes by all hosts attached to that segment. Ethernet cards have a filter that prevents the host machine from seeing traffic addressed to other stations. Sniffing programs turn off the filter, and thus see everyone's traffic.

   Today's networks are increasingly employing "switch" technology, preventing this technique from being as successful as in the past. It is still useful, though, as it is becoming increasingly easy to install remote sniffing programs on servers and routers, through which a lot of traffic flows.

   Today's networks may already contain built-in sniffing modules. Most hubs support the RMON standard, which allow the intruder to sniff remotely using SNMP, which has weak authentication. Many corporations employ Network Associates "Distributed Sniffer Servers", which are set up with easy to guess passwords. Windows NT machines often have a "Network Monitoring Agent" installed, which again allows for remote sniffing.

   A packet sniffing is very difficult to detect. Stealing Passwords is related to being able to retrieve the password of some different user using a service and accessing the service on that user's behalf. It may deal with the stealing of one user's password.

   To avoid such password attacks the password need to be protected from the place of storage to the way of transfer. The following are some methods which helps the protection of password.

**1.2 Storing password:**

1. Users authentication information is individually configured into every server the user will use.
2. Another location called authentication storage node, stores user information and servers retrieve that information when they want to authenticate the user.
3. Another location is called as authentication facilitator node. This location stores the user information and a server that

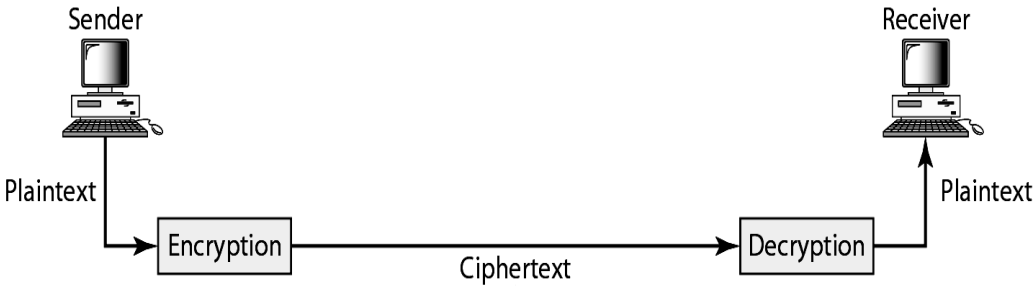
wants to authenticate user will send the information received from the user to the AF node.

4. **Encrypted password:**
   Normally with any networking system the password from the node to the server will always travel in a plain text format. In the same way within the server database also the password will be stored in a plain text format. It will be easy for any hacker to crack the password. To avoid this kind of attack one should keep the password in an encrypted format. Any encryption method can be used to encrypt the password.

## 2. Cryptographic based authentication:

### 2.1 Introduction to Basic Encryption and Decryption:

The term 'Cryptography' means the concept of encryption and decryption together. Cryptography is the technique in which the original 'plain text' message is 'encrypted' i.e. converted into a coded form called 'cipher text' at the sender's end, which is then transmitted to the receiver. The receiver then 'decrypts' i.e. converts the 'cipher text' back into the 'plain text' to get the original message back.




Cryptography is also called as an art or technique to achieve secure communication between the communicating parties by encoding the messages between them such that no third party can gain anything useful out of interception.

Various techniques are utilized for this purpose of cryptography. Broadly these techniques fall into two categories.

1) Symmetric key cryptography: - in which the 'key' element used, is the 'same' for both encryption as well as decryption and
2) Asymmetric key cryptography - in which the 'key' element used, is different for both encryption as well as decryption.

   - Symmetric key cryptography is also known as 'private or secret key cryptography'
     Whereas

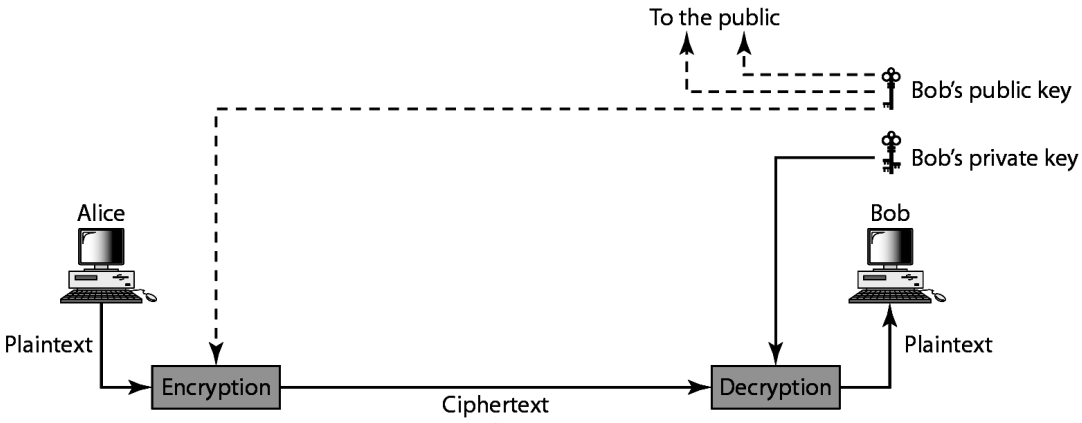
- Asymmetric key cryptography is also known as 'public key cryptography', (*please refer to section 5.6 of these notes for details*)

The techniques used in symmetric key cryptography are as below.

## 2.2 Asymmetric Key Cryptography:

In public-key cryptography, there are two keys: a private key and a public key. The receiver keeps the private key. The public key is announced to the public.

Imagine Alice, as shown in Figure 29.20, wants to send a message to Bob. Alice uses the public key to encrypt the message. When Bob receives the message, the private key is used to decrypt the message.




In public-key encryption/decryption, the public key that is used for encryption is different from the private key that is used for decryption.

The public key is available to the public; the private key is available only to an individual.

## 2.2 Protocols:

Various protocols are used for providing the authentication mechanism.

**Authentication protocols overview:**

Authentication is a fundamental aspect of system security. It confirms the identity of any user trying to log on to a domain or access network resources. The server authentication enables single sign-on to all network resources. With single sign-on, a user

can log on to the domain once, using a single password or smart card, and authenticate to any computer in the domain.

**Authentication types:**

When attempting to authenticate a user, several industry-standard types of authentication may be used, depending on a variety of factors.

| Authentication protocols | Description |
|---|---|
| Kerberos V5 authentication | A protocol that is used with either a password or a smart card for interactive logon. It is also the default method of network authentication for services. |
| SSL/TLS authentication | A protocol that is used when a user attempts to access a secure Web server. |
| NTLM authentication | A protocol that is used when either the client or server uses a previous version of Windows. |
| Digest authentication | Digest authentication transmits credentials across the network as an MD5 hash or message digest. |
| Passport authentication | Passport authentication is a user-authentication service which offers single sign-in service. |

All the above mentioned types will be discussed in detail in the various sections of this chapter.

**2.3 The advantages & disadvantages of key techniques:**

**Public-key encryption/decryption has two advantages.**

First, it removes the restriction of a shared symmetric key between two entities (e.g., persons) that need to communicate with each other. A shared symmetric key is shared by the two parties and cannot be used when one of them wants to communicate with a third party. In public-key encryption! Decryption, each entity creates a pair of keys; the private one is kept, and the public one is distributed. Each entity is independent, and the pair of keys created can be used to communicate with any other entity.

**The second advantage** is that the number of keys needed is reduced tremendously.

In this system, for I thousand users to communicate, only 1 thousand pairs of keys i.e. 2000 keys are needed, not 4,99,500, as was the case in symmetric-key cryptography.

**Public-key cryptography also has two disadvantages.**

The big disadvantage is the complexity of the algorithm. If we want the method to be effective, the algorithm needs large numbers. Calculating the cipher text from plaintext using the long keys takes a lot of time. That is the main reason that public-key cryptography is not recommended for large amounts of text.

**Public-key algorithms are more efficient for short messages.**

The second disadvantage of the public-key method is that the association between an entity and its public key must be verified. If Alice sends her public key via an email to Bob, then Bob must be sure that the public key really belongs to Alice and nobody else.

One point needs to re-mention that if your private key were made public you would Get Bankrupted in no time!

## 3. Address based authentication

### 3.1. KDCs

One way to make things manageable is to use a trusted node known as Key Distribution Center (KDC).The KDC knows keys for all nodes. If a new node is installed in the network, only that, new node and the KDC need to be configured with a key for that node. If node α wants to talk to node β, α talks to the KDC(securely ,since α and the KDC share a key),and asks for a key with which to talk to β .The KDC authenticates α ,chooses a random number Rαβ to be used as a key to be shared by α and β for their conversation ,encrypts Rαβ with the key the KDC shares with α and gives that to α. The KDC also encrypts Rαβ with the key the KDc shares with β and gives that to β. With the instruction that it is to be used for conversing with α.(Usually ,the KDC will not bother to actually transmit the encrypted Rαβ to β but rather will give it to α to forward to β.)the encrypted message to β that the KDC gives to α to forward is often referred to as a ticket. Besides containing Rαβ, the ticket generally contains other information such as an expiration time and α's name. KDCs make key distribution much more convenient .when a new user is being installed into the network ,or when a user's key is suspected of having been compromised ,there's a single location(the KDC) that needs to be configured .The alternative to using a KDc is installing the user's information at

every server to which the user might access. There are some disadvantages to KDCs, though:

- The KDC has enough information to impersonate anyone to anyone .If it is compromised, all the network resources are vulnerable.
- The KDC is a single point of failure .If it goes down, nobody can use anything on the network (or rather, nobody can start using something on the network-keys previously distributed can continue to be used).It is possible to have multiple KDC's which share the same database of keys, but that means added complexity and cost for extra machines and replication protocols, and added vulnerability, since there are now more targets that need to be protected.
- The KDC might be a performance bottleneck, since everyone will need to frequently communicate with it. Having multiple KDCs can alleviate this problem.

**Certificate Revocation:**

The PKI method distributes the certificates using third parties. These certificated are providing the additional security mechanism to the existing message exchange. For certificate status to be determined, public key infrastructure (PKI) certificate revocation information must be made available to individuals, computers, network devices, and applications attempting to verify the validity of certificates. Traditionally, a PKI uses a distributed method of verification so that the clients do not have to contact the Certification Authority (CA) directly to validate the credentials presented. Instead, clients connect to alternate resources, such as Web servers or Lightweight Directory Access Protocol (LDAP) directories, where the CA has published its revocation information. Without checking certificates for revocation, the possibility exists that an application or user will accept credentials that have been revoked by a CA administrator.

Certificates are issued with a planned lifetime, which is defined through a validity start time and an explicit expiration date. For example, a certificate may be issued with a validity of one day, thirty years, or even longer. Once issued, a certificate becomes valid when its validity time has been reached, and it is considered valid until its expiration date. However, various circumstances may cause a certificate to become invalid prior to the expiration of the validity period. Such circumstances include change of name (for example, requiring to change the subject of a certificate due to an employee's change of name), change of association between subject and CA (for example, when an employee terminates employment with an organization), and compromise or suspected

compromise of the corresponding private key. Under such circumstances, the issuing CA needs to revoke the certificate.

There are several mechanisms to represent revocation information. RFC 3280 defines one such method. This method involves each CA periodically issuing a signed data structure called a certificate revocation list (CRL). A CRL is a list identifying revoked certificates, which is signed by a CA and made freely available at a public distribution point. The CRL has a limited validity period, and updated versions of the CRL are published when the previous CRL's validity period expires. Each revoked certificate is identified in a CRL by its certificate serial number. When certificate-enabled software uses a certificate (for example, for verifying a remote user's digital signature), the software should not only check the certificate signature and time validity, but it should also acquire a suitably recent certificate status to ensure that the certificate being presented is not revoked. Normally, a CA will automatically issue a new CRL on either a configured, regular periodic basis (for example, daily or weekly), or the CRL can be published manually by a CA administrator

There is a potential disadvantage with CAs .Suppose Fred is given a certificate with an expiration time a year in the future and then Fred is fired .Since Fred is now a disgruntled ex-employee, it would be nice to alert the network not to allow him access .With KDCs, it is easy-merely delete his key from the KDC. With CAs, though it is not as straight forward to deny access to someone once he is given a certificate. It is common practice to put an expiration date in a certificate .The certificate becomes invalid after that day. The typical validity interval is about a year. A disgruntled ex-employee can do a lot of damage in a year, even without a machine gun. But you wouldn't want validity intervals much smaller than that, because renewing certificates is a nuisance.

The solution is similar to what was done for credit cards. When the bank issues a credit card, it prints an expiration date, perhaps a year in the future. But sometimes a card is reported is stolen, or the bank might for some other reason like to revoke it. The credit card company publishes a book of credit card numbers that stores should refuse to honour .these days, most stores are hooked to computer networks where they check the validity of the card. But in ancient times, merchants needed to rely on the hook of that credit card number, which was presumably published frequently.

## 3.3 BIOMETRICS

Biometrics are the another way to ensure the security mechanism. This authenticates the user by verifying either the one

which the user possesses or the one which user has as a physical features. The one which the user possesses may be a smart card or SID chip. The one which the user has may be his fingerprints or the facial expressions. There are variety of Biometrics devices available. All are too expensive to be in everyday use, but in some cases the costs are coming down to where we may see these. Technology available today includes:

- **Retinal Scanner.** This is device that examines the tiny blood vessels in the back of your eye. The layout is as distinctive as a fingerprint and apparently easier to read. These devices are quite expensive and have a “psychologically threatening” user interface.

- **Fingerprint readers**. This would seem an obvious technology since fingerprints have been used as a method of identification for many years. For some reason, automating this technology has never been very successful, though there are devices available.

- **Face recognition.** Looking at a digitized picture of a person, a computer can measure facial dimensions and do a good job of recognizing people just don’t show up at work with a black eye and a swollen jaw.

- **IRIS scanner.** Like a retinal scanner, this maps the distinctive layout of the iris of your eye. It has the major advantage of having a less intimidating user interface- rather than requiring you to look into a laser device; iris scans can be done with a camera several feet away and might even be done covertly.

- **Handprint readers.** These are more widely used than fingerprint readers. They measure the dimensions of the hand: finger length, width, and so on. They aren’t as accurate as fingerprints, but they are less expensive and less problem-prone.

- **Voiceprints.** It turns out that it’s possible to do a frequency spectrum analysis of someone’s eyes and get identification nearly as accurate as a fingerprint. This technology is in use, but has not caught on in spite of the fact that it should be fairly cheap. It can be defeated with a tape recording, and it may refuse to authenticate someone whose voice has shifted due to illness.

- **Keystroke timing**. The exact way in which people type is quite distinctive and experiments have been done with identification based on the way people type. There is a

problem that various injuries can throw off timing, and the networks that connect terminals and computers tend to loose the keystroke timing information before it reaches a processor that can use it.

- **Signatures.** There is a classic human form of authentication, and there are human experts quite adept at determining whether two signatures were produced by the same person. Machines thus far have not been able to duplicate that ability. However, when not just the signature is recorded, but the actual timing of the moments that go into scribing the signature, there is sufficient information for authentication and some systems use this method, with the user signing on an electronic tablet.

## ADDRESS - BASED AUTHENTICATION

Address-based authentication doesn't rely on sending passwords around the network, but rather assumes that the identity of the source can be inferred based on the network address from which packets arrive. It was adopted early in the evolution of computer networks by both UNIX and VMS.

On UNIX, the Berkeley rtools support such access; on VMS, similar functionality is called PROXY. The general idea can be implemented in various ways.

- Machine B might have a list of network addresses of "equivalent" machines. If machine A is listed, then any account name on A is equivalent to the same account name on B. If a request from A arrives with the name John Smith, then the request will be honored if it for anything that the account John Smith on B was allowed to do. This has the problem that the user has to have the identical account name on all systems.

- Machine B might instead have a list of <address, remote account name, local account name>. If a request arrives from address A with name Jekyll, then the database is scanned for the matching entry, say <A, Jekyll, Hyde>. Then the request is honored provided the local account Hyde is authorized to do the request.
  UNIX implements two account mapping schemes:

- /etc/hosts.equiv file. There is a global file which implements the first scheme above. The file/ etc/hosts.equiv on a machine A contain a list of computers that have identical user account assignments. Suppose a computer B is listed

in /etc/hosts.equiv. Suppose A receives a request with account name Smith, and B's address in the source address field of the network header. If an account with the name smith exists in machine A, the request will be given the same privileges as the local user Smith. The /etc/hosts.eqiv file is useful in managing corresponding accounts in bulk on machines with common accounts and common management.

- Per-user .rhosts files. In each UNIX user's home directory, there can be a file named .rhosts, which contains a list of <computer, account> pairs that are allowed access to the user's account. Any user Bob can permit remote access to his account by creating a .rhosts file is his home directory. The account names need not be the same on the remote machine as they are on this one, so Bob can handle the case where he has different account names on different systems. Because of the way the information is organized, any request that is not for an account named the same as the source account must include the name of the account that should process the request. For instance, if the local account name on system A is Bob and the request is from computer B, account name Smith, the request has to specify that it would like to be treated with the privileges given account name Bob.

  There is a centrally managed proxy database that says for each remote computer <computer, account> pair what account(s) that pair may access, usually with one of them marked as the default. For example, there might be an entry specifying that account Smith from address B should have access to local accounts Bob and Alice, where the account Bob might be marked as the default.

  The VMS scheme makes access somewhat more user-friendly than the UNIX scheme in the case where a user has different account names on different systems. Generally (in VMS) the user needn't specify the target account in that case. In the rare case, where a user is authorized to access multiple accounts on the remote computer a target account can be specified in the request to access an account other than the default.

  Address-based authentication is safe from eavesdropping.

**Verification techniques:**

**Passwords:**

Today's networks are increasingly employing "switch" technology, preventing this technique from being as successful as in the past. It is still useful, though, as it is becoming increasingly easy to install remote sniffing programs on servers and routers, through which a lot of traffic flows.

Today's networks may already contain built-in sniffing modules. Most hubs support the RMON standard, which allow the intruder to sniff remotely using SNMP, which has weak authentication. Many corporations employ Network Associates "Distributed Sniffer Servers", which are set up with easy to guess passwords. Windows NT machines often have a "Network Monitoring Agent" installed, which again allows for remote sniffing.

A packet sniffing is very difficult to detect. Stealing Passwords is related to being able to retrieve the password of some different user using a service and accessing the service on that user's behalf. It may deal with the stealing of one user's password.

Thus the attacker can do the following:

- Use the resources of the user if its the system password
- Use the service if it's a web service password
- Use the financial resources of the user, if the password is of some financial institution etc.

- A password stealer's main motto is to get unauthorized access into a genuine user's account and use the service illegally.

- But this activity may take a major advancement if the stealer happens to access the entire password file for all users of a given system or service. This means that the passwords of many different users cab be misused.

- Usually, passwords are simple. Either they are day-to-day usage terms or some words with which the user is associated. For an attacker keen on stealing password, a simple brute force attack, covering different words from the dictionary is enough.

Thus, necessary precautions have to be taken in order to avoid important passwords from getting stolen. We should be careful that we take into account the following major points of consideration.

- Never reveal our password to anybody. Especially, when that same password coincides with the password of some different service, which is very important.
- Password files should be placed separately in a different server (Password server) and not be a part of the Web Server.
- Passwords should be a mixture of alphabets, numbers and special symbols such that it is not a part of the daily usage dictionary, which can be detected through brute force attack.
- As network administrators a minimum of say 3 attempts must be kept for a user entering wrong password. This will cut down the chances of brute force to a greater extent.

Various methods to prevent from password stealing:

**1. One time password:**

This is the basic method, which will use different password with every access. As the passwords are easily cracked by the systems, this method will help the user to protect the password from the hackers. Every time the password will change. Before the hackers use any method to crack the password, the user will change the password.

**2. Encrypted password:**

Normally with any networking system the password from the node to the server will always travel in a plain text format. In the same way within the server database also the password will be stored in a plain text format. It will be easy for any hacker to crack the password. To avoid this kind of attack one should keep the password in an encrypted format. Any encryption method can be used to encrypt the password.

## 3.4 Smart Cards

### 3.4.1. Introduction:

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

Organizations are steadily migrating toward this technology. The days are numbered for a single mainframe used for computing every directive. Today, the delegation of tasks is being transferred to small, but dedicated smart cards. Their usefulness may soon

exceed that of the standard computer for a variety of applications due, in part, to their portability and ease of use.

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

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

### 3.4.2 Architecture:

1. Most smart cards have been designed with the look and feel of a credit or debit card, but can function on at least three levels (credit - debit - personal information). Smart cards include a microchip as the central processing unit, random access memory (RAM) and data storage of around 10MB.

2. The smart card is an electronic recording device. Information in the microchip can instantaneously verify the cardholder's identity and any privileges to which the cardholder may be entitled. Information such as withdrawals, sales, and bills can be processed immediately and if/when necessary; those records can be transmitted to a central computer for file updating.

3. Smart cards are secure, compact and intelligent data carriers. Smart cards should be regarded as specialized computers capable of processing, storing and safeguarding thousands of bytes of data.

4. Smart cards have electrical contacts and a thin metallic plate just above center line on one side of the card. Beneath this dime-sized plate is an integrated circuit (IC) chip containing a central processing unit (CPU), random access memory (RAM) and non-volatile data storage.

5. Data stored in the smart card's microchip can be accessed only through the chip operating system (COS), providing a high level of data security. This security takes the form of passwords allowing a user to access parts of the IC chip's memory or encryption/decryption measures which translate the bytes stored in memory into useful information.

6. Smart cards typically hold 2,000 to 8,000 electronic bytes of data (the equivalent of several pages of data). Because those bytes can

be electronically coded, the effective storage capacity of each card is significantly increased.

7. Magnetic-stripe cards, such as those issued by banks and credit card companies, lack the security of microchips but remain inexpensive due to their status as a single-purpose card.

8. Smart cards can be a carrier of multiple records for multiple purposes. Once those purposes are maximized, the smart card is often viewed as superior and, ultimately, less expensive.

9. The distributed processing possible with smart cards reduces the need for ever-larger mainframe computers and the expense of local and long-distance phone circuits required to maintain an on-line connection to a central computer.

10. Smart cards are defined by the ISO 7816 standards.

### 3.4.3. Security aspects:

1. The microprocessor on the smart card is there for **security**. The host computer and card reader actually "talk" to the microprocessor.

2. The microprocessor enforces access to the data on the card. If the host computer read and wrote the smart card's random access memory(RAM), it would be no different than a diskette..

3. Smarts cards may have up to 8 kilobytes of RAM, 346 kilobytes of ROM 256 kilobytes of programmable ROM, and a 16-bit microprocessor.

4. The smart card uses a serial interface and receives its power from external sources like a card reader. The processor uses a limited instruction set for applications such as cryptography.

### 3.4.4. Applications:

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

## 3.4.5.Types of Smart Cards

### Contact Cards and Contactless Cards

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

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

### Categories of Smart Cards

- Integrated Circuit (IC) Microprocessor Cards: Allow for adding, deleting, or manipulating information in memory, allowing for a variety of applications and dynamic read/write capabilities. Most Smart Cards in use for mobile applications are of this type.

- IC Memory Cards: Can store data, but do not have a processor on the card.

- Optical Memory Cards: Can only store data, but have a larger memory capacity than IC memory cards.

### QUESTIONS:

1. Explain the need of passwords
2. What are the various methods to store the password
3. Write a short note on certificate revocation
4. What are the various applications of smart cards.
5. Write in detail about the functions and advantages of smart cards
6. Write in detail about the various ways to implement biometrics
7. Write a short note on address based authentication
8. Write a short note on cryptographic based authentication
9. What are the various authentication protocols
10. Explain in detail about the key technology used with authentication.

# 6

# SECURITY POLICIES & SECURITY HANDSHAKE PITFALLS

**In this unit, we are going to learn about the following topics:**

- What is security policy
- High and low level policy, user issues
- Protocol problems
- Assumptions
- Shared secret protocols
  - ✓ Diffie-Hellman Algorithm
- Public key protocols
- Mutual authentication
- Reflection attacks
- Use of timestamps
- Nonce and sequence numbers
- Session keys
- One-and two-way public key based authentication.
  - ✓ Hash Function
  - ✓ Collection
  - ✓ Properties
  - ✓ 2Way Authentication

## 6.1. Security Policy:

**Definition:**

- A security policy is the set of decisions that collectively, determines an organization's attitude toward security.
- **A security policy defines the boundaries of acceptable behavior and what the response to violations should be.**

Security policy is the set of decisions/rules & regulation written or verbally understood that collectively determines Organization's posture towards security. It delimits the boundaries, specifies acceptable & non acceptable behaviors, dictates what is ethical and what is non-ethical states the degree of seriousness of the offence and also mention the consequences of the actions if it is violated. It

focuses on different stack holders and satisfying their requirements in an efficient way.

**Organization differs in their** Culture, Structures and Strategy. Thus Security Policy will also differ from organization to organization.

**Security Policy will decide on the following issues.**

- What legal course of action will you follow if attacked?
- What will be considered as a cognizable crime?
- Can anyone be sued?
- Infringing on someone else's rights?

**To devise a security policy you must yourself several question?**

1. What resources are you trying to protect?
2. Who would be interested in attacking you?
3. How much security can you afford?

Your security policy may determine what legal course you have to take if you are ever attacked. You must first decide what is and is not permitted. To some extent, this process is driven by the business or structural needs of the organization. Thus, some companies may issue a verdict that bars the personal' use of corporate computers. Some companies wish to restrict outgoing traffic, to guard against employees exporting valuable data. Other policies may be driven by technological considerations.

In general, **Computer security** means keeping anyone from doing anything, which is unwanted or undesired, relating to computers & peripherals. It is the way of protecting your precious assets in terms of information or resources.

### 6.1.1. Picking a security policy:

A 'Security Policy' describes your plan, methodology to safeguard your assets or what measures / precautions you take (or do not take) in order to keep your assets secured. A security policy differs from organization to organization. All the decisions are then based on this formulated policy. The first step here is to perform a **Risk Analysis**. It is a process of examining all your risks & then finding a cost-effective decision to recover from it. A few important steps in this are:

1. **Finding out what resources you wish to protect:** Resources may include: Physical resources like printers, monitors, keyboards, drives, modems etc. & Logical resources like source & object programs, data, utilities, operating system, applications etc.

   What resources are you trying to protect? The answer to this will dictate the host specific measures that are needed. Machines with sensitive files may require extra security measures: Stronger authentication, keystroke logging and strict auditing, or even file encryption. If the target of interest is the outgoing connectivity, the administrator may choose to require certain privileges for access to the network. May be all such access should be done through a proxy that will perform extra logging.

2. **Find out who can disrupt them & in what ways:** The threats to your assets may include
   - Physical threats to the resources such as stealing, malfunctioning devices,
   - Logical threats such as unauthorized access to data, information, resources
   - Unintended disclosure of your information.

3. **Who is interested in attacking you?**
   - Outsiders as well as insiders may form the collective answer here.
   - What kind of security therefore must be provided differs from the type of attacker you are planning against.

**4. How much Security can you afford?**

Part of the cost of security is direct financial expenditures, such as the extra routers, firewalls, software packages, and so on. Often, the administrative costs are overlooked. There is another cost, however, a cost in convenience and productivity, and even moderate. Too much security can hurt as surely as too little can. Annoyed by increases in security, people get frustrated. Finding the proper balance therefore is essential.

**5. How to decide on the user levels and the privileges? What Stance do you take?**

The stance is the attitude of the designer. It is determined by the cost of failure and the designer’s estimate of that likelihood. It is also based on the designer's opinions of their own abilities. At one end of the scale is a philosophy to correct it only when mistake happens and the other one is taking preventive measures so that no mistake occurs.

### 6.1.2. Experiment in Picking a Security Policy:

**What is a policy should I have while surfing?**

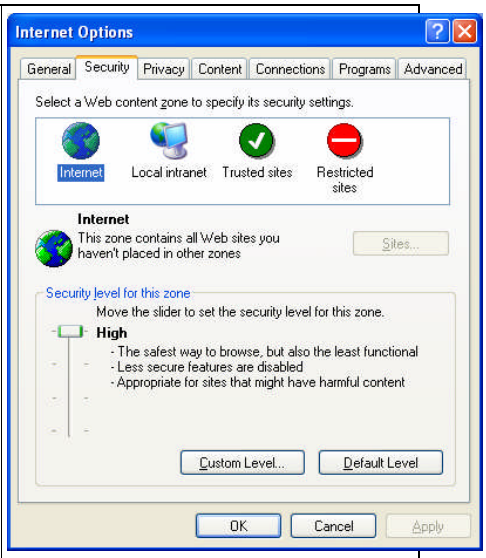
**Let us do a practical experiment**

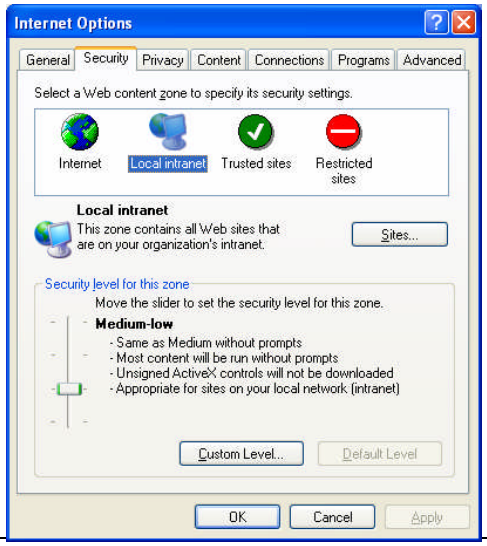
1. **Open Internet Explorer**
2. **Click on "tool"**
3. **Click on "Internet Options"**
4. **Click on "Security"**
5. **Click on "Custom Level"**
6. **Click on "Default levels"**

**Study the following levels.**

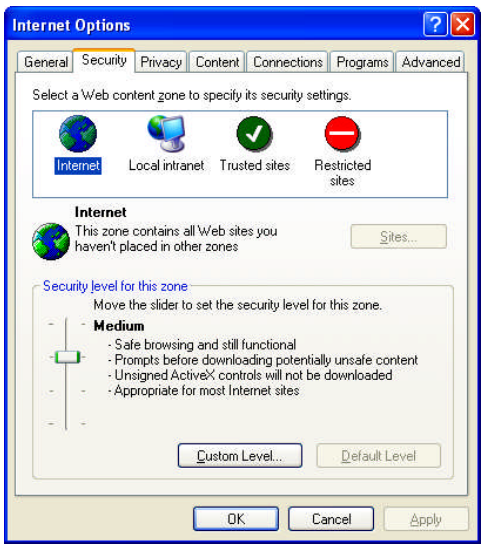
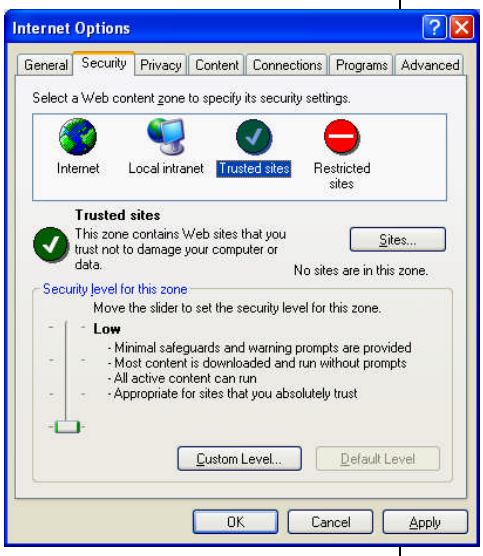
1. **High**
2. **Medium**
3. **Medium-low and**
4. **Low**

**High:**
The safest way to browse,
but also the least functional
Less secured feature are disabled
Appropriate policy for sites
that might have harmful contents




**Medium-Low**
Same as Medium but with prompts
Most content will run without prompts
Unsigned ActiveX controls will be downloaded
Appropriate for sites on local Network (Intranets)

<table>
<tr>
<td> 
</td>
<td>Medium<br>Safe browsing & still functional<br>Prompts before downloading potentially unsafe content<br>Unsigned ActiveX controls will be downloaded<br>Appropriate for most Internet sites</td>
</tr>
<tr>
<td>Low<br>Minimam safeguards and warning promts provided<br>Most content is downloaded and run without prompts<br>All active contents can run<br>Approariate for site that you absolutely trust.</td>
<td> 
</td>
</tr>
</table>

The problems reported by the organization may be from different areas. The problems that affect the privacy and security of the organization are to be resolved. To resolve them, one has to aware of those problems to analyze the scenario completely. Let us discuss about those problems first.

### 6.1.3. Problems with protocols:

Sometimes, the protocol used in the networks also has certain limitations or problems contained in them, which prevent the applications from doing the appropriate things. Since they work from behind the applications, this may increase the vulnerability.

An example of such failure is the TCP protocol failure. TCP provides the circuits or paths for the IP datagrams. These may be sent across the network. The attackers checking for the packets can get information about the source IP. Similarly the IP is a stateless and unreliable protocol. No guarantee of delivery of packets can be given for it. It is possible for attackers to send packets using any known or valid source address. This is called source address spoofing. Although the operating system controls this, still it cannot be relied on.

All the classes of attacks discuss situations in which everything was working properly, but trustworthy authentication was not possible. Here, in Protocol failures, we consider the reverse: i.e. areas where the protocols themselves are inadequate, thus denying the application the opportunity to do the right thing.

Secure protocols must rest on a secure foundation. Consider SSH, which is a fine protocol for secure remote access. SSH has a feature where a user can specify a trusted public key by storing it in a file called authorized keys (local file). Then, if the client knows the private key, the user can log in without having to type a password. IN UNIX, this file typically resides in the .ssh directory in the user's home directory. Now, consider the case in which someone uses the Ssh to log into a host, an attacker can spoof the replies to inject a fake authorized keys file.

The authorized keys file introduces another type of vulnerability. If a user gets a new account in a new environment he typically copies all of the important files there from an existing account, including the .ssh directory, so that all of the .ssh keys are available from the new account. However, the user may not realize that copying the authorized keys file means that this new account can be accessed by any key trusted to access the previous account.

### 6.1.4. Symmetric-key algorithms/shared secret key protocols:

**Symmetric-key algorithms** are a class of algorithms for cryptography that use trivially related, often identical, cryptographic keys for both decryption and encryption. The encryption key is trivially related to the decryption key, in that they may be identical or there is a simple transformation to go between the two keys. The keys, in practice, represent a shared secret between two or more parties that can be used to maintain a private information link. Other terms for symmetric-key encryption are **secret-key**, **single-key**, **shared-key**, **one-key**, and **private-key** encryption. Use of the last and first terms can create ambiguity with similar terminology used in public key cryptography.

### 6.1.5. Secret Key Exchange

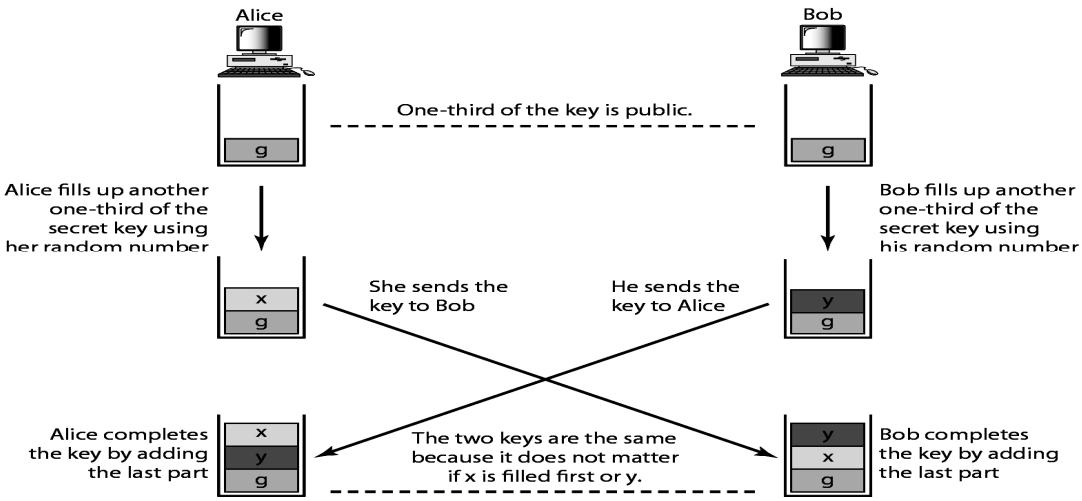
For symmetric key cryptography to work for online communications, the secret key must be securely shared with authorized communicating parties and protected from discovery and use by unauthorized parties. Public key cryptography can be used to provide a secure method for exchanging secret keys online. Two of the most common key exchange algorithms are the following:

- Diffie-Hellman Key Agreement algorithm
- RSA key exchange process

Both methods provide for highly secure key exchange between communicating parties. An intruder who intercepts network communications cannot easily guess or decode the secret key that is required to decrypt communications. The exact mechanisms and algorithms that are used for key exchange varies for each security technology. In general, the Diffie-Hellman Key Agreement algorithm provides better performance than the RSA key exchange algorithm.

### 6.2 Diffie-Hellman Key exchange algorithm:

Whitefield Diffie and Martin Hellman, in 1976 have come out with a good solution to the problem of key distribution as mentioned above. The steps of this algorithm are as given below.




### Steps for algorithm:

Assume two parties viz. 'first' and 'second' want to communicate securely.

1. Let 'first' and 'second' agree upon two large prime nos., say n and g. These need not be kept secured. (i.e. everyone can know these values.)

2. 'First' chooses another large random no. say x to calculate another number A such that, A = g^xmod n. (Note, value of x is only known to 'first'!)
3. This no. A is then sent by 'first' to 'second'.
4. 'Second' also chooses another large random no. say y to calculate another number B such that,
5. B = g^y mod n. (Note, value of y is only known to 'second'!)
6. This no. B is then sent by 'second' to 'first'.
7. Now, independently, 'first' calculates the key KI as: KI = B^xmod n
8. Also, 'second' independently calculates the key K2 as: K2 = A^y mod n
9. As it should be required here in symmetric key cryptography, KI = K2.

**Example:**

Let us take an actual example, to illustrate above algorithm.

Assuming values such as n= 11, g=7, x=3 and y=6, we have following equations:

1. Value of A=7^3mod 11 =343 mod 11 =2.
2. Value of B = 7^6mod 11 = 117649 mod 11 = 4.
3. Key KI = 4^3mod 11 = 64 mod 11 = 9.
4. And, Key K2 = 2^6mod 11 = 64 mod 11 = 9.
5. Thus, we find that KI = K2.
6. Hence the algorithm is proved.

### 6.2.1. The public-key protocol

This assumes the use of a public key algorithm. Here, Alice (A) and Bob (B) use a trusted server (S) to distribute public keys on request. These keys are:

- **$K_{PA}$ and $K_{SA}$, respectively public and private halves of an encryption key-pair belonging to A**
- **$K_{PB}$ and $K_{SB}$, similar belonging to B**
- **$K_{PS}$ and $K_{SS}$, similar belonging to S. (Note this has the property that $K_{SS}$ is used to *encrypt* and $K_{PS}$ to *decrypt*).**

The protocol runs as follows:

$$A \rightarrow S : A, B$$

**A requests B's public keys from S**

$$S \rightarrow A : \{K_{PB}, B\}_{K_{SS}}$$

**S responds with public key $K_{PB}$ alongside B's identity, signed by the server for authentication purposes.**

$$A \rightarrow B : \{N_A, A\}_{K_{PB}}$$

**A invents $N_A$ and sends it to B.**

$$B \rightarrow S : B, A$$

B requests A's public keys.

$$S \rightarrow B : \{K_{PA}, A\}_{K_{SS}}$$

Server responds.

$$B \rightarrow A : \{N_A, N_B\}_{K_{PA}}$$

B invents $N_B$, and sends it to A along with $N_A$ to prove ability to decrypt with $K_{SB}$.

$$A \rightarrow B : \{N_B\}_{K_{PB}}$$

A confirms $N_B$ to B, to prove ability to decrypt with **$K_{SA}$.** At the end of the protocol, A and B know each other's identities, and know both $N_A$ and $N_B$. These nonces are not known to eavesdroppers.

### 6.2.2. An attack on the protocol

Unfortunately, this protocol is vulnerable to a man-in-the-middle attack. If an impostor I can persuade A to initiate a session with him, he can relay the messages to B and convince B that he is communicating with A. Ignoring the traffic to and from S, which is unchanged, the attack runs as follows:

$$A \rightarrow I : \{N_A, A\}_{K_{PI}}$$

A sends $N_A$ to I, who decrypts the message with **$K_{SI}$**

$$I \rightarrow B : \{N_A, A\}_{K_{PB}}$$

I relay the message to B, pretending that A is communicating

$$B \rightarrow I : \{N_A, N_B\}_{K_{PA}}$$

B sends **$N_B$**

$$I \rightarrow A : \{N_A, N_B\}_{K_{PA}}$$

I relay it to A

$$A \rightarrow I : \{N_B\}_{K_{PI}}$$

A decrypts $N_B$ and confirms it to I, who learns it

$$I \rightarrow B : \{N_B\}_{K_{PB}}$$

I re-encrypt $N_B$, and convince B that he's decrypted it. At the end of the attack, B falsely believes that A is communicating with him, and that $N_A$ and $N_B$ are known only to A and B.

**6.2.3. Security mechanisms can be combined / embedded with protocols:**

- **Timestamp**
- **Nonce**
- **Sequence Numbers**

**1. Timestamp and its usage:**

A **timestamp** is a sequence of characters, denoting the date and/or time at which a certain event occurred. A timestamp is the time at which an event is recorded by a computer, not the time of the event itself. In many cases, the difference may be inconsequential: the time at which an event is recorded by a timestamp (e.g., entered into a log file) should be very, very close to the time of the occurrence of the event recorded. This data is usually presented in a consistent format, allowing for easy comparison of two different records and tracking progress over time; the practice of recording timestamps in a consistent manner along with the actual data is called **time stamping**.

Timestamps are typically used for logging events, in which case each event in a log is marked with a timestamp. In file systems, timestamp may mean the stored date/time of creation or modification of a file.

**Assumptions:**

Every timestamp value is unique and accurately represents an instant in time. No two timestamps can be the same. A higher-valued timestamp occurs later in time than a lower-valued timestamp.

**Examples of timestamps:**

2005-10-30 T 10:45 UTC
2007-11-09 T 11:20 UTC
Sat Jul 23 02:16:57 2005
1256953732

**2.Nonce and Sequence Numbers:**

In security engineering, a **nonce** is an abbreviation of **number used once** (it is similar in spirit to a nonce word). It is often a random or pseudo-random number issued in an authentication protocol to ensure that old communications cannot be reused in replay attacks. For instance, nonces are used in HTTP digest access authentication to calculate a MD5 digest of the password. The nonces are different each time that the 401 authentication challenge response code is presented, and each client request has a unique sequence number, thus making replay attacks and dictionary attacks virtually impossible.Some also refer to initialization Vectors as nonce for the above reasons. To ensure that a nonce is used only once, it should be time-variant (including a suitably granular timestamp in its value), or generated with enough random bits to ensure a probabilistically insignificant chance of repeating a previously generated value. Some authors define pseudo-randomness (or unpredictability) as a requirement for a nonce.

**3 .Mutual authentication** or **two-way authentication:**

**Mutual authentication or two-way authentication** (sometimes written as 2WAY authentication) refers to two parties authenticating each other suitably. In technology terms, it refers to a client or user authenticating themselves to a server and that server authenticating itself to the user in such a way that both parties are assured of the others' identity. When describing online authentication processes, mutual authentication is often referred to as website-to-user authentication, or site-to-user authentication. Typically, this is done for a client process and a server process without user interaction.

Mutual SSL provides the same things as SSL, with the addition of authentication and non-repudiation of the client authentication, using digital signatures. However, due to issues with complexity, cost, logistics, and effectiveness, most web applications are designed so they do not require client-side certificates. This creates an opening for a man-in-the-middle attack, in particular for online banking.

**Mutual Authentication** is a security feature in which a client process must prove its identity to a server, and the server must prove its identity to the client, before any application traffic is sent over the client-to-server connection. Support for mutual authentication is provided by the security support provider interface (SSPI) and is exposed directly through the SSPI APIs and services that layer upon SSPI, including RPC and COM+.

Not all security packages available to SSPI, nor all services running Windows 2000, support mutual authentication. An application must request mutual authentication and a supporting security package to obtain mutual authentication.

Mutual authentication requires that the client and server prove their respective identities to each other before performing any application functions. Identity can be proved through a trusted third party and use shared secrets, as in Kerberos v5, or through cryptographic means, as with a public key infrastructure. Each party is identified by a **principal name.**

Secure Shell (SSH) public key authentication can be used by a client to access servers, if properly configured. These notes describe how to configure OpenSSH for public key authentication, how to enable a ssh-agent to allow for passphrase-free logins, and tips on debugging problems with SSH connections. Password free logins benefit remote access and automation.

Public key authenticate can prevent brute force SSH attacks, but only if all password-based authentication methods are disabled. Other options to protect against brute force SSH attacks include pam_tally, or port knocking. Public key authentication does not work well with Kerberos **or** OpenAFS, which require a password or principal from the client.

**Definition of terms used:**

- ***Client***: the system one types directly on, such as a laptop or desktop system.
- ***Server***: anything connected to from the client. This includes other servers accessed through the first server connected to.

***Note: Never allow root-to-root trust between systems.*** If required by poorly engineered legacy scripts, limit the access of the public keys, and if possible only allow specific public keys to run specific commands. Instead, setup named accounts for users or roles, and grant as little- root access as possible via pseudo.

### 6.2.4. Reflection Attack:

It is a method of attacking a challenge-response authentication system that uses the same protocol in both directions. That is , the same challenge-response protocol is used by each side to authenticate the other side. The essential idea of the attack is to trick the target into providing the answer to its own challenge.

The general attack outline is as follows:

1) The attacker initiates a connection to a target.
2) The target attempts to authenticate the attacker by sending it a challenge.
3) The attacker opens another connection to the target, and sends the target this challenge as its own.
4) The target responds to the challenge.
5) The attacker sends that response back to the target on the original connection.

If the authentication protocol is not carefully designed, the target will accept that response as valid; thereby leaving the attacker with one fully-authenticated channel connection and the other one is simply abandoned.

Some of the most common solutions to this attack are described below:

The responder sends its identifier within the response so, if it receives a response that has its identifier in it, it can reject it.

1. Alice initiates a connection to Bob.
2. Bob challenges Alice by sending a nonce.B->A:N
3. Alice responds by sending back her identifier and the nonce encrypted using the shared key Kab. A→B:{A,N}Kab
4. Bob decrypts the message, makes sure that it is from Alice and not a message he had sent in the past by finding A in it and not B and if the nonce is the same as the one he sent in his challenge, then he accepts the message.
5. But, it requires the initiating party to first respond to challenges before the target party responds to its challenges.

**Require the key or Sequence Numbers**

A fundamental notion in the design is that every octet of data sent over a TCP connection has a sequence number. Since every octet is sequenced, each of them can be acknowledged. The acknowledgment mechanism employed is cumulative so that an acknowledgment of sequence number X indicates that all octets up to but not including X have been received. This mechanism allows for straight-forward duplicate detection in the presence of retransmission. Numbering of octets within a segment is that the first data octet immediately following the header is the lowest numbered, and the following octets are numbered consecutively. t

is essential to remember that the actual sequence number space is finite, though very large. This space ranges from 0 to 2**32 - 1. Since the space is finite, all arithmetic dealing with sequence numbers must be performed modulo 2**32. This unsigned arithmetic preserves the relationship of sequence numbers as they cycle from 2**32 - 1 to 0 again. There are some subtleties to computer modulo arithmetic, so great care should be taken in programming the comparison of such values. The symbol "=<" means "less than or equal" (modulo 2**32).

**The typical kinds of sequence number comparisons which the TCP must perform include:**

(a) Determining that an acknowledgment refers to some sequence number sent but not yet acknowledged.

(b) Determining that all sequence numbers occupied by a segment have been acknowledged (e.g., to remove the segment from a retransmission queue).

(c) Determining that an incoming segment contains sequence numbers which are expected (i.e., that the segment "overlaps" the receive window). In response to sending data the TCP will receive acknowledgments.

The following c**omparisons** are needed to process the acknowledgments.

| **Sequence Signal** | **Description** |
|---|---|
| **SND.UNA** | oldest unacknowledged sequence number |
| **SND.NXT** | next sequence number to be sent |
| **SEG.ACK** | acknowledgment from the receiving TCP (next sequence number expected by the receiving TCP) |
| **SEG.SEQ** | first sequence number of a segment |
| **SEG.LEN** | the number of octets occupied by the data in the segment (counting SYN and FIN) |

| | |
|---|---|
| **SEG.SEQ+SEG.LEN-1** | last sequence number of a segment |
| **RCV.NXT** | next sequence number expected on an incoming segments, and is the left or lower edge of the receive window |
| **RCV.NXT+RCV.WND-1** | last sequence number expected on an incoming segment, and is the right or upper edge of the receive window |
| **SEG.SEQ** | first sequence number occupied by the incoming segment |
| **SEG.SEQ+SEG.LEN-1** | last sequence number occupied by the incoming segment |

### 6.2.5. Session Key:

A **session key** is a single-use symmetric key used for encrypting all messages in one communication session. A closely related term is **traffic encryption key** or **TEK**, which refers to any key used to encrypt messages as opposed to different uses, such as encrypting other keys (key encryption key)

Session keys introduce complication in a crypto system, normally to an undesirable end. However, they also help with some real problems, which is why they are used. There are two primary reasons for session keys:

- First, several cryptanalytic attacks are made easier as more material encrypted with a specific key is available. By limiting the material processed using a particular key; those attacks are made more difficult.

- Second, asymmetric encryption is too slow for general purpose use, but many otherwise good encryption algorithms require that keys be distributed securely before encryption can be used. All secret key algorithms have this undesirable property. By using asymmetric encryption to distribute a secret key for another, faster, symmetric algorithm, it's possible to improve overall performance considerably. Like all cryptographic keys, session keys must be chosen so that they are unpredictable by an attacker. In the usual case, this means that they must be chosen randomly. Failure to choose session keys (or any key) properly is a major (and too common in actual practice) design flaw in any crypto system.

**One-Way and Two-Way public key based authentication:**

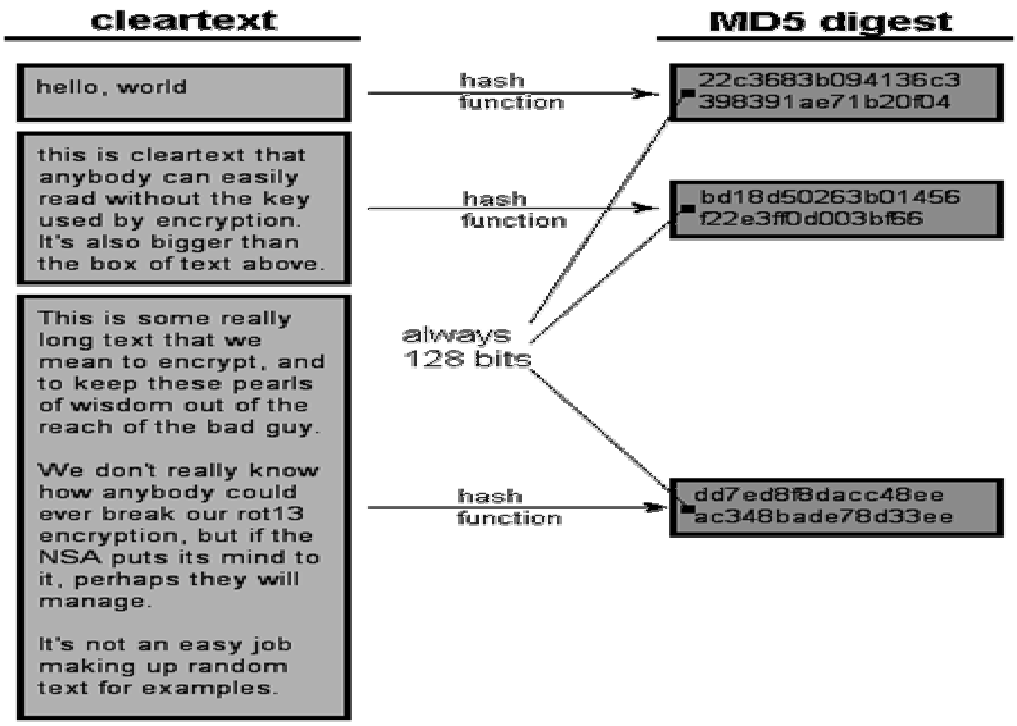
### 6.2.6. One-way Hash Function:

A **cryptographic hash function** is a deterministic procedure that takes an arbitrary block of data and returns a fixed-size bit string, the (**cryptographic**) **hash value**, such that an accidental or intentional change to the data will change the hash value. The data to be encoded is often called the "message", and the hash value is sometimes called the **message digest** or simply **digests**.

The ideal cryptographic hash function has four main properties:

- it is easy to compute the hash value for any given message,
- it is infeasible to find a message that has a given hash,
- it is infeasible to modify a message without changing its hash,
- it is infeasible to find two different messages with the same hash.

Cryptographic hash functions have many information security applications, notably in digital signatures, message authentication codes (MACs), and other forms of authentication. They can also be used as ordinary hash functions, to index data in hash tables, for fingerprinting, to detect duplicate data or uniquely identify files, and as checksums to detect accidental data corruption. Indeed, in information security contexts, cryptographic hash values are sometimes called (**digital**) **fingerprints**, **checksums**, or just **hash values**, even though all these terms stand for functions with rather different properties and purposes.




## 6.3. Applications of hash functions:

### Hash tables:

Hash functions are primarily used in hash tables, to quickly locate a data record (for example, a dictionary definition) given its search key (the headword). Specifically, the hash function is used to map the search key to the hash. The index gives the place where the corresponding record should be stored. Hash tables, in turn, are used to implement associative arrays and dynamic sets.

In general, a hashing function may map several different keys to the same index. Therefore, each slot of a hash table is associated with (implicitly or explicitly) a set of records, rather than a single record. For this reason, each slot of a hash table is often called a **bucket,** and hash values are also called **bucket indices**.

Thus, the hash function only hints at the record's location — it tells where one should start looking for it. Still, in a half-full table, a good hash function will typically narrow the search down to only one or two entries.

### 1. Caches

Hash functions are also used to build caches for large data sets stored in slow media. A cache is generally simpler than a

hashed search table, since any collision can be resolved by discarding or writing back the older of the two colliding items.

### 2. Bloom filters

Hash functions are an essential ingredient of the Bloom filter, a compact data structure that provides an enclosing approximation to a set of keys.

### 3. Finding duplicate records

To find duplicated records in a large unsorted file, one may use a hash function to map each file record to an index into a table *T*, and collect in each bucket *T*[*i*] a list of the numbers of all records with the same hash value *i*. Once the table is complete, any two duplicate records will end up in the same bucket. The duplicates can then be found by scanning every bucket *T*[*i*] which contains two or more members, fetching those records, and comparing them. With a table of appropriate size, this method is likely to be much faster than any alternative approach (such as sorting the file and comparing all consecutive pairs).

### 4. Finding similar records

Hash functions can also be used to locate table records whose key is similar, but not identical, to a given key; or pairs of records in a large file which have similar keys. For that purpose, one needs a hash function that maps similar keys to hash values that differ by at most *m*, where *m* is a small integer (say, 1 or 2). If one builds a table of *T* of all record numbers, using such a hash function, then similar records will end up in the same bucket, or in nearby buckets. Then one need only check the records in each bucket *T*[*i*] against those in buckets *T*[*i*+*k*] where ***k* ranges between -*m* and *m*.**

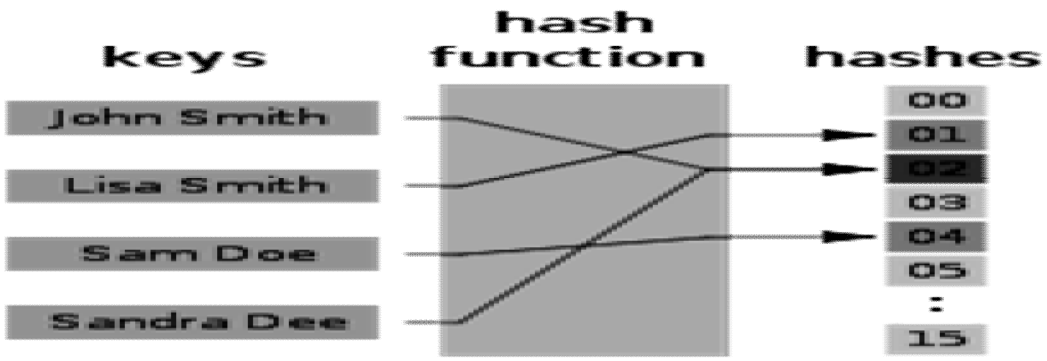
### 5. Finding similar substrings

The same techniques can be used to find equal or similar stretches in a large collection of strings, such as a document repository or a genomic database. In this case, the input strings are broken into many small pieces, and a hash function is used to detect potentially equal pieces, as above.

### 6. Geometric hashing

This principle is widely used in computer graphics, computational geometry and many other disciplines, to solve many proximity problems in the plane or in three-dimensional space, such as finding closest pairs in a set of points, similar

shapes in a list of shapes, similar images in an image database, and so on. In these applications, the set of all inputs is some sort of metric space, and the hashing function can be interpreted as a partition of that space into a grid of ***cell**s*. The table is often an array with two or more indices, and the hash function returns an index tuple. This special case of hashing is known as geometric hashing or **the grid method**. Geometric hashing is also used in telecommunications to encode and compress multi-dimensional signals.




### 6.3.1. Properties of Hash Functions:

Good hash functions, in the original sense of the term, are usually required to satisfy certain properties listed below.

**1. Low cost**

The cost of computing a hash function must be small enough to make a hashing-based solution advantageous with regard to alternative approaches. For instance, binary search can locate an item in a sorted table of $n$ items with $\log_2 n$ key comparisons. Therefore, a hash table solution will be more efficient than binary search only if computing the hash function for one key cost less than performing $\log_2 \boldsymbol{n}$ key comparisons. One advantage of hash tables is that they do not require sorting, which keeps the cost of the hash function constant regardless of the rate at which items are added to the data set.

**2. Determinism**

A hash procedure must be deterministic — meaning that for a given input value it must always generate the same hash value. In other words, it must be a function of the hashed data, in the mathematical sense of the term. This requirement excludes hash functions that depend on external variable parameters, such as pseudo-random number generators that depend on the time of day. It also excludes functions that depend on the memory address

of the object being hashed, if that address may change during processing although sometimes rehashing of the item can be done.

**3. Uniformity**

A good hash function should map the expected inputs as evenly as possible over its output range. That is, every hash value in the output range should be generated with roughly the same probability. The reason for this last requirement is that the cost of hashing-based methods goes up sharply as the number of **collisions** — pairs of inputs that are mapped to the same hash value — increases. Note that this criterion only requires the value to be **uniformly distributed**, not *random* in any sense. A good randomizing function is usually good for hashing, but the converse need not be true.

Hash tables often contain only a small subset of the valid inputs. For instance, a club membership list may contain only a hundred or so member names, out of the very large set of all possible names. In these cases, the uniformity criterion should hold for almost all typical subsets of entries that may be found in the table, not just for the global set of all possible entries. When testing a hash function, the uniformity of the distribution of hash values can be evaluated by the chi-square test.

**4. Variable range**

In many applications, the range of hash values may be different for each run of the program, or may change along the same run (for instance, when a hash table needs to be expanded). In those situations, one needs a hash function which takes two parameters — the input data *z*, and the number *n* of allowed hash values. A common solution is to compute a fixed hash function with a very large range divide the result by *n*, and use the division's remainder. If *n* itself a power of 2, this can be done by bit masking and bit shifting. When this approach is used, the hash function must be chosen so that the result has fairly uniform distribution between 0 and $n-1$, for any *n* that may occur in the application. Depending on the function, the remainder may be uniform only for certain *n*, e.g. odd or prime numbers.

**5. Data normalization**

In some applications, the input data may contain features that are irrelevant for comparison purposes. For example, when looking up a personal name, it may be desirable to ignore the distinction between upper and lower case letters. For such data, one must use a hash function that is compatible with the data equivalence criterion being used: that is, any two inputs that

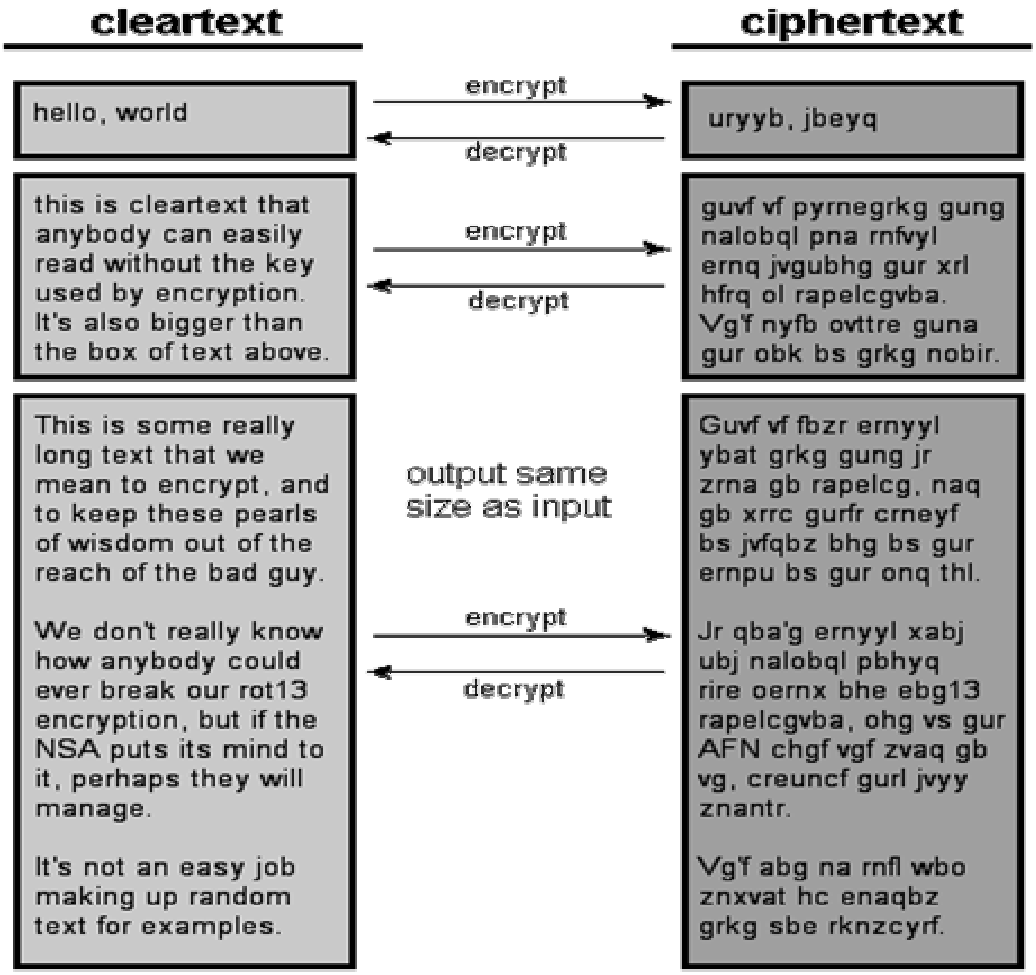
are considered equivalent must yield the same hash value. This can be accomplished by normalizing the input before hashing it, as by upper-casing all letters.

**6. Continuity**

A hash function that is used to search for similar (as opposed to equivalent) data must be as continuous as possible; two inputs that differ by a little should be mapped to equal or nearly equal hash values. Note that continuity is usually considered a fatal flaw for checksums, cryptographic hash.

**7. Two-way public key based authentication:**

Mutual authentication and two-way authentication is one and the same and we discussed about mutual authentication in the earlier sessions. The diagram below depicts the process of two-way authentication.



**Questions for practice:**

1. What is security Policy? Explain the meaning of security policy? What are the questions to be asked while framing a security policy?
2. How security policy can be used for organizational effectiveness towards security?
3. What are the problems faced by the secret-key exchange algorithms? Explain.
4. State and explain the Diffie-Hellman Key exchange algorithm.
5. “The protocols, rules for the communication, may lead to further problem in client-server terminology”-Explain this statement.
6. Explain the concept of public- key protocols in detail. Also state the possible attacks on the protocol.
7. What is mutual authentication? How 2Way authentication is achieved in cryptography? Explain.
8. Name and explain the attack is in challenge-response authentication. Also suggest some solutions for the same.
9. Explain the requirement of sequence numbers in authentication.
10. Explain some of the applications of hash functions.
11. State and explain the properties of the hash functions.
12. What is one-way authentication? How it is achieved in cryptography? Explain.
13. Compare and contrast one-way and two-way public key based authentication.
14. Write a short note on the following:
    - Nonce
    - Session keys
    - Time Stamps

**Computer Security Terms**

- Ciphertext - Text or data that has been enciphered such that unauthorized persons should be unable to read it.
- Decipher - The act of taking text that is enciphered so unauthorized persons cannot read it and processing it into plain text or data that can be read or used.

- Encipher - The act of taking plain text or data and processing it into ciphertext so unauthorized personnel cannot read it or use it.
- MAC - Message Authentication Code
- MDC - Manipulation Detection Code
- MIC - Message Integrity Check
- One way hash functions - A function that can be easily performed one way but is difficult or impossible to reverse.
- Plaintext - Normal text or data, before the enciphering process or after the deciphering process.
- Salting - Adding random data to a password prior to performing a one way hash on it. This is a means of stopping a dictionary attack.

# 7

# EXAMPLE SYSTEM

1. Kerberos:
    1. Purpose
    2. Authentication
    3. Server and ticket granting server
    4. keys and tickets, use of ASS and TGS, replicated servers
2. Kerberos V4:
    1. Names,
    2. Inter-realm authentication,
    3. key version numbers
3. Kerberos V5:
    1. Names. Realms,
    2. Delegation, forwarding and proxies
    3. Ticket lifetimes, revoking tickets, multiple realms

## 1. Kerberos:

### 1.1 Purpose:

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

**The Internet is an insecure place.** Many of the protocols used in the Internet do not provide any security. Tools to "sniff" passwords off of the network are in common use by malicious hackers. Thus, applications which send an unencrypted password over the network are extremely vulnerable. Other applications rely on the client to restrict its activities to those which it is allowed to do, with no other enforcement by the server.

Some sites attempt to use firewalls to solve their network security problems. Unfortunately, firewalls assume that the hackers

are always from the outside, which is often a very bad assumption. Most of the really damaging incidents of computer crime are carried out by insiders. Firewalls also have a significant disadvantage in that they restrict the usage of internet by the users.

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

The first published repot on Kerberos listed the following requirements;

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

**Two versions:**
Version 4
Version 5

**1.2 Keys:**

**1. Keys: Privacy through Encryption**

Kerberos messages are encrypted with various encryption keys to ensure that no one can tamper with the client's ticket or with other data in a Kerberos message.
Possible Kerberos keys include:

***2. Long-term key***

This is a key which is known only to the target server and the KDC — with which the client's ticket is encrypted.

***3. Client/server session key***

A short-term, single-session key that is created by the KDC and used to encrypt the client-to-server and server-to-client messages after identity and authorization have been confirmed.

***4. KDC/user session key***

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

**1.3 Four parties:**

Alice ---- he client workstation

Authentication Server AS) ------- Authenticates the user during login

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

Bob ------ he server offering services such as network printing. File sharing or an application program.

TGS is to certify to the servers in the network that a user is really what he claims to be.

**1.4 How Kerberos works**

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

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

**1.4.1 Kerberos Encryption**

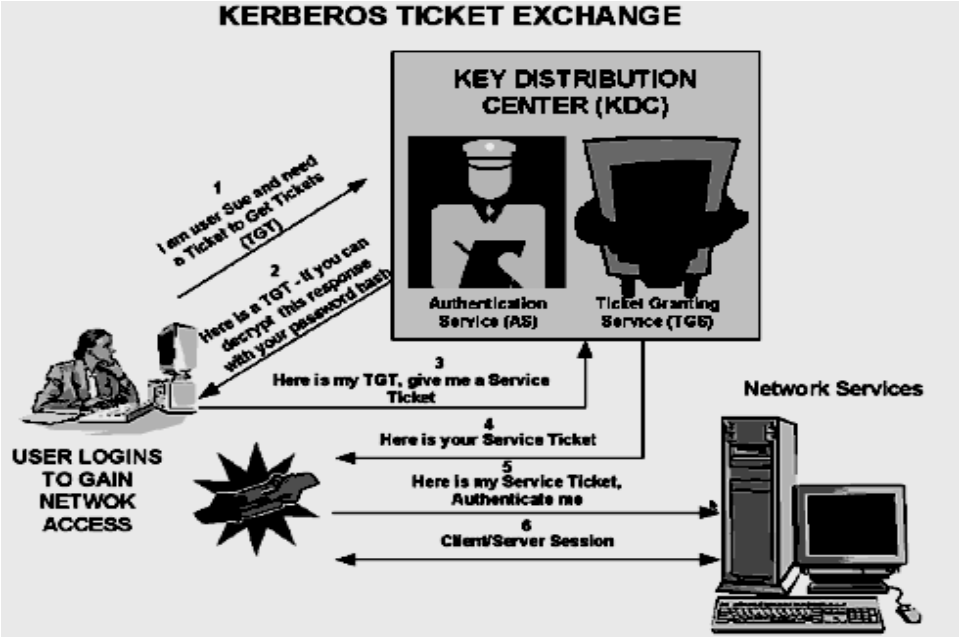
Though conceptually, Kerberos authentication proves that a client is running on behalf of a particular user, a more precise statement is that the client has knowledge of an encryption key that is known by only the user and the authentication server. In Kerberos, the user's encryption key is derived from and should be thought of as a password; we will refer to it as such in this article. Similarly, each application server shares an encryption key with the authentication server; we will call this key the server key.

Encryption in the present implementation of Kerberos uses the data encryption standard (DES). It is a property of DES that if cipher text (encrypted data) is decrypted with the same key used to encrypt it, the plaintext (original data) appears. If different encryption keys are used for encryption and decryption, or if the cipher text is modified, the result will be unintelligible, and the checksum in the Kerberos message will not match the data. This combination of encryption and the checksum provides integrity and confidentiality for encrypted Kerberos messages.

## 1.5 The Kerberos Ticket

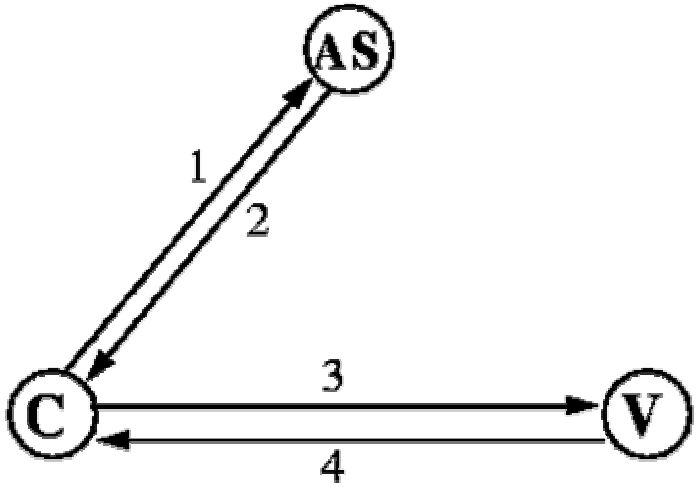
The client and server do not initially share an encryption key. Whenever a client authenticates itself to a new verifier it relies on the authentication server to generate a new encryption key and distribute it securely to both parties. This new encryption key is called a *session key* and the Kerberos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

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



### 1.5.1 Application request and response

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




1. as_req: c, v, $time_{exp}$, n
2. as_rep: $\{K_{c,v}, v, time_{exp}, n, ...\}K_c$, $\{T_{c,v}\}K_v$
3. ap_req: $\{ts,ck, K_{subsession}, ...\}K_{c,v}$ $\{T_{c,v}\}K_v$
4. ap_rep: $\{ts\}K_{c,v}$ (optional)

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

**Figure 1: Basic Kerberos authentication protocol (simplified)**

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

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

### 1.6 Authentication request and response

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

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

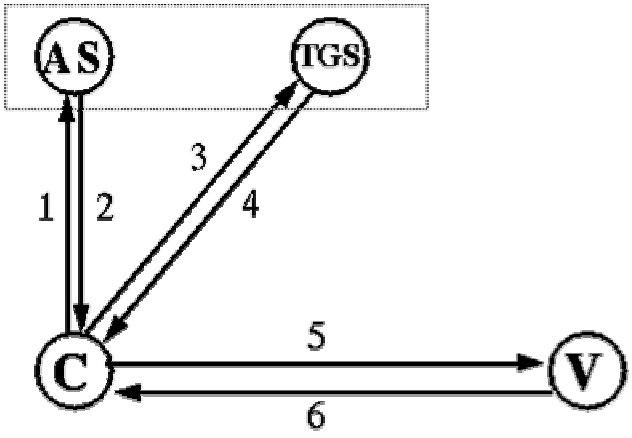
#### 1.6.1 Obtaining additional tickets

The basic Kerberos authentication protocol allows a client with knowledge of the user's password to obtain a ticket and session key for and to prove its identity to any verifier registered with the authentication server. The user's password must be presented each time the user performs authentication with a new verifier. This can be cumbersome; instead, a system should support single sign-on, where the user logs in to the system once, providing the password at that time, and with subsequent authentication occurring automatically. The obvious way to support this, caching the user's password on the workstation, is dangerous. Though a Kerberos ticket and the key associated with it are valid for only a short time, the user's password can be used to obtain tickets, and to impersonate the user until the password is changed. A better approach, and that used by Kerberos, is to cache only tickets and encryption keys (collectively called credentials) that will work for a limited period.

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

Subsequently, when the user wishes to prove its identity to a new verifier, a new ticket is requested from the authentication server using the ticket granting exchange. The ticket granting exchange is identical to the authentication exchange except that the ticket granting request has embedded within it an application request, authenticating the client to the authentication server, and the ticket granting response is encrypted using the session key from the ticket granting ticket, rather than the user's password.

Figure 2 shows the complete Kerberos authentication protocol. Messages 1 and 2 are used only when the user first logs in to the system, messages 3 and 4 whenever a user authenticates to a new verifier, and message 5 is used each time the user authenticates itself. Message 6 is optional and used only when the user requires mutual-authentication by the verifier.




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

**Figure 2: Complete Kerberos Authentication Protocol (simplified)**

## 2 Kerberos Version 4:

### 2.1. Key concepts:

**KDC (Key Distribution Centre):**

This is physically securing the node with complete authentication database. The KDC shares a secret key, known as a master key, with each principal. When Alice informs the KDC that she wants to talk to Bob, the KDC invents a session key Kab for Alice and Bob to share, encrypts Kab with Alice's master key for Alice, encrypts Kab with Bob's master key for Bob, and returns all this information to Alice. The message consisting of the session key Kab encrypted with Bob's master key is known as a ticket to Bob. The session key Kab together with the ticket to Bob are known as Alice's credentials to Bob.

**1. Ticket Granting Ticket(TGT):**

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

**2. Ticket Granting Server (TGS):**

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

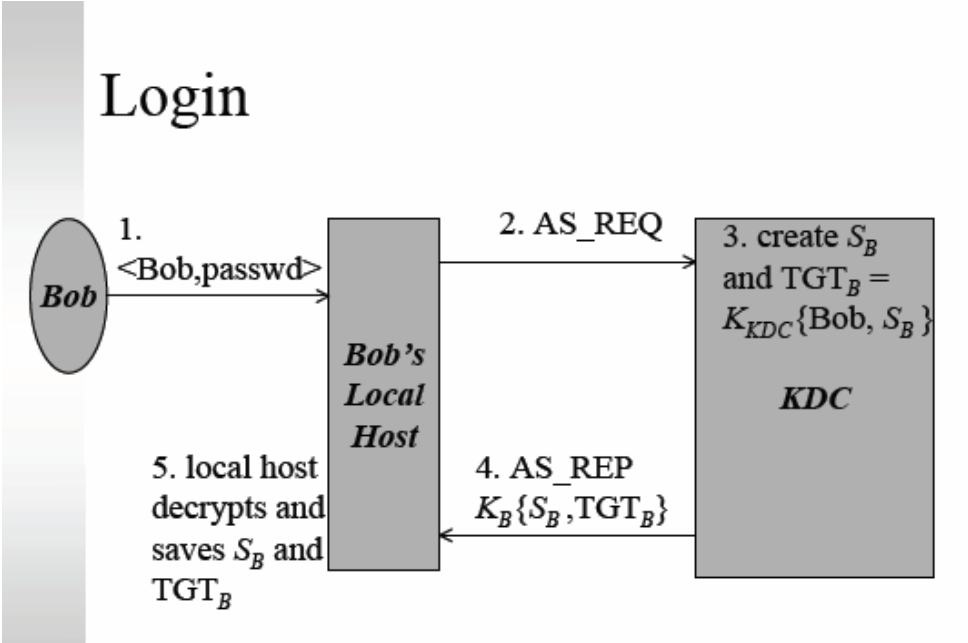
**3. Session Key and Ticket-granting Ticket (TGT)**

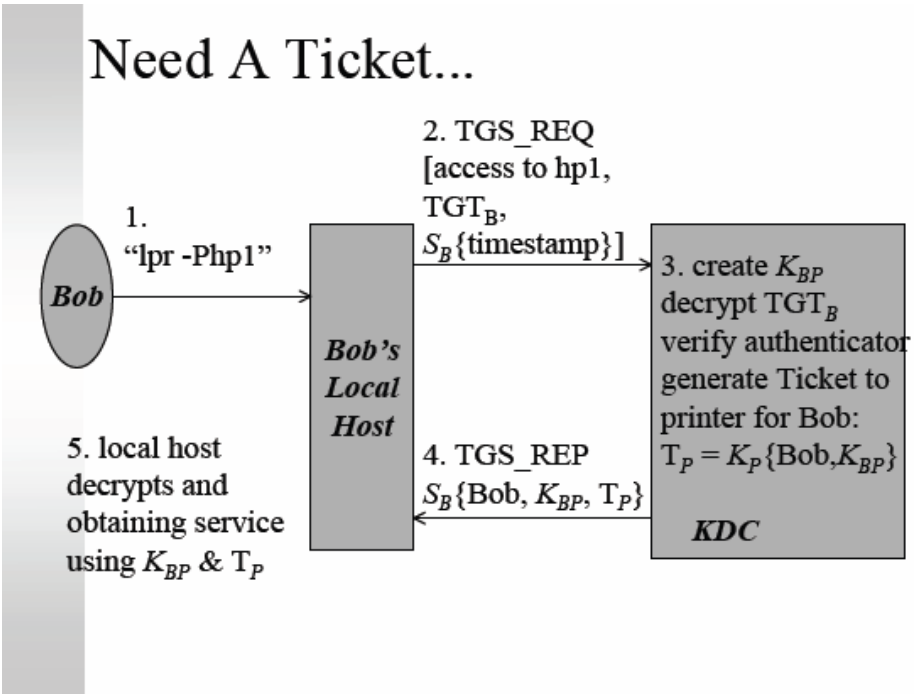
The messages between a host and the KDC can be protected using the principal's master key. For every request to KDC from the principal needs the session key. It insists on principal retyping in the password. It is important to remember the principal's password and the principal's master key derived from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

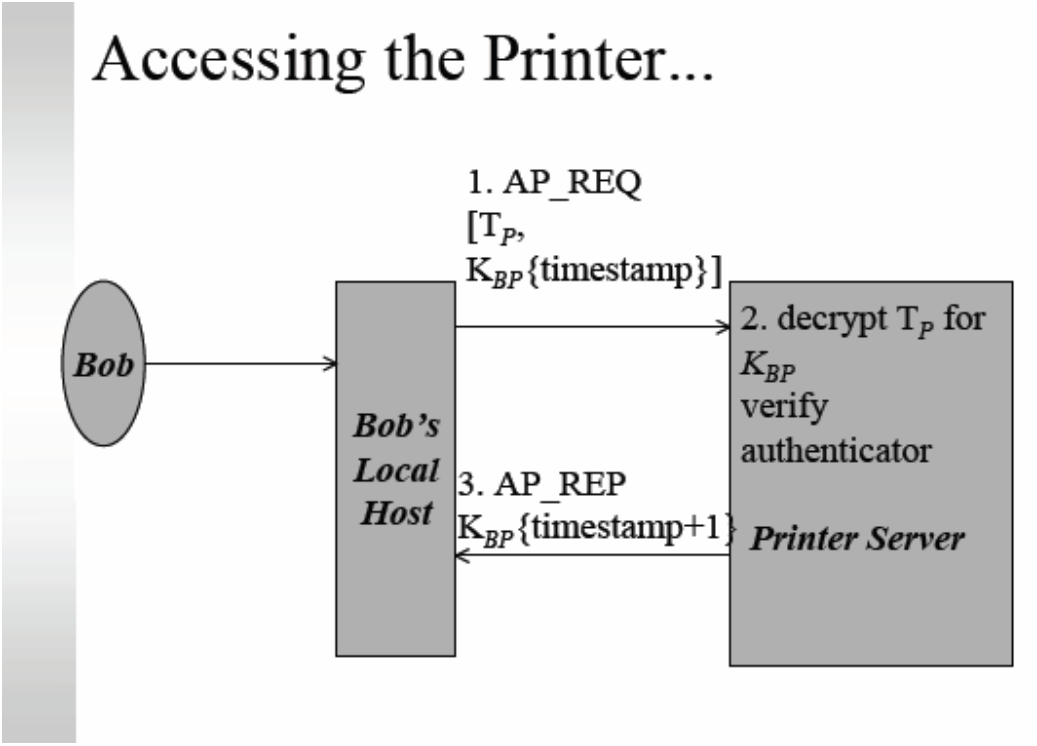
To avoid potentially too much exposure to password/master key it is necessary at initial login, a per principal session key *SB* (for Bob) is requested from KDC.*SB* has a limited valid time period. A TGT for Bob is also issued by the KDC, which includes the session key *SB* and Bob's identification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

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

***Login:***






**2.2 Realms:**

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

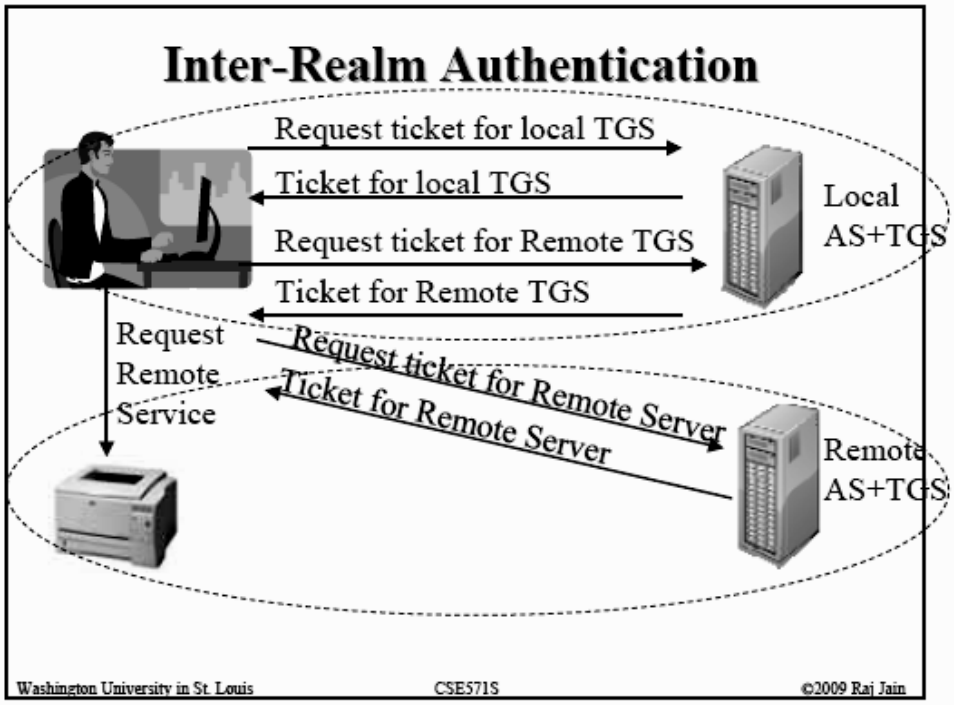
Because of this reason the network are divided into realms. Each and every realm has its own KDC database. There can be multiple KDC's in a realm, but they would be equivalent, have the same KDC, aster key, and have the same database of principal's master keys. Two KDC/s in different realms, would have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

**2.3 Key components:**

1. Realm = One organization or one trust domain
2. Each realm has its own set of principles including KDC/TGT
3. Each Principal's name = Name + Instance + Realm

4. Each principle is of 40 characters each with null terminated.
5. Instance = Particular Server or Human role (administrator, game player)
6. In V4, both realms should have a direct trust relationship and chaining prohibited

**Interrealm authentication:**




## 3. Kerberos V5:

### 3.1Name:

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

### 3.2 Delegation of rights:

One way delegation could be provided is for Alice to send bob her master key (in an encrypted message to bob.), allowing him to obtain tickets to whatever resources he might need on Alice behalf. That is clearly not desirable, since it would allow bob forever

be able to impersonate Alice. These mechanisms are inconvenient and /or insecure and therefore undesirable but they wouldn't work with Kerberos (either V4 or V5). Kerberos V5 explicitly allows delegation by allowing Alice to ask for TGT with a network layer address different from hers. As a matter of fact it allows Alice to specify multiple addresses to include (in which case the ticket can be used form any of the specified addresses), or allows Alice to request that no address be included (in which case the ticket can be used for any address.) Kerberos V5 supports two forms of limited delegations:

1. Alice can give Bob tickets to the specific services he will need to access on her behalf (rather than giving him a TGT, which will allow him to request tickets to any services.)

2. When requesting a ticket or TGT that she intends to give Bob, Alice can request that the field AUTHORIZATION-DATA be added to the ticket or TGT.

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

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

## 3.3 Ticket Lifetime:

In Kerberos V5, tickets can be issued with virtually unlimited life times. The time stamp format is an ASN.1- defined quantity that is 17 octets long. Although it has a virtually unlimited lifetime, it is only in seconds, and Kerberos V5, in some cases, would have

preferred time expressed down to micro-seconds. Long- lived tickets pose serious security risk, because once created they cannot be revoked. So V5 has a number of mechanisms for implementing revocable long-lived tickets. These mechanisms involve use of several timestamp fields in tickets.

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

### 3.3.1. Renewable Tickets:

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

### 3.3.2 Postdated Tickets:

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

occurs, Alice can present the ticket to the KDC and the KDC will clear the **invalid** flag. This additional step gives the opportunity to revoke the postdated ticket by warning the KDC. If the KDC is configured to revoke the postdated ticket, the validation request will fail.

**QUESTIONS:**

1. Write a short note on Kerberos
2. What are the various types of Kerberos
3. Explain the concept of interrealm authentication with Kerberos
4. Give an example for the Kerberos system.
5. What are the various types of tickets with Kerberos V5.
6. Explain about the various keys used with Kerberos.

# 8

# NETWORK SECURITY

1. **Email security**
   - SMTP: (Simple Mail Transfer Protocol)
   - PGP :(Pretty Good Privacy)
   - PEM (Privacy Enhanced Mail)
2. **IP security**
   - Introduction
   - Security Architecture
   - Encapsulating Security Payload
3. **Network Management security**
   - Overview
   - Management Information Base (MIB)
   - Security

## 1. Electronic mail security:

Nowadays the electronic mail transactions are getting the importance because any user can send message to another user which may contain video, music, pictures etc. The wide usage of Email system has increased the need of security with the same. An email message consists of two parts:

1) Contents (OR) body
2) Headers

Various protocols are used with email transaction to provide the security services such as:

1. Origin Authentication
   Making sure you know who the message came from.
2. Content Integrity
   Ensuring the message has not been altered after being sent.

3. Non-Repudiation
   Making sure the originator cannot deny sending a message at a later date
4. Encryption
   Encoding the content of the message to prevent unauthorized reading.

The following protocols are supporting the secured email transaction.

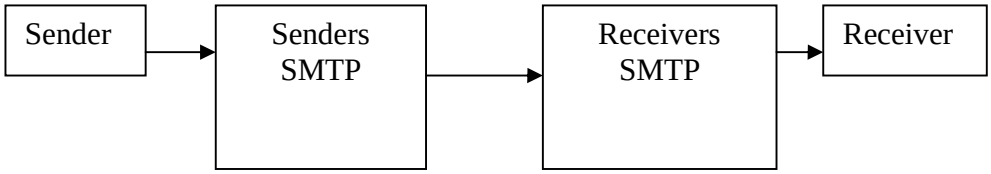
SMTP(Simple Mail Transfer Protocol)

PEM (Privacy Enhanced Mail)

PGP (Pretty Good Privacy)

**1.1SMTP: (Simple Mail Transfer Protocol)**

This SMTP is mainly used for email communications. The sender's end gives the email message to the local SMTP server. This server in turn transfers the message to the receiver's SMTP server. This SMTP protocol works on top of TCP/IP.




**Three steps:**

1. Sender to Sender SMTP
2. Sender SMTP to receiver's SMTP
3. Receiver's SMTP to receiver.

**Characteristics:**

1. It is very simple architecture.
2. It uses human understandable ASCII test format.
3. The receiver's end can access the SMTP server by using any other simple email protocols like Post Office Protocol (POP), Internet Mail Access Protocol (IMAP) etc.

**Detailed steps:**

1. Based on the client's request, the server sends a Ready for Mail (RFM) reply, indicating that it can accept an email message from the client.
2. The client then sends a Hello command to start the session.

3. The server then sends back an acknowledgement by sending its own DNS name.
4. The client now sends one or more email messages to the server.
5. The email transfer begins with a MAIL command.
6. The recipient allocates buffers to store the incoming email message and sends back an OK response to the client.
7. The client now sends the list of the intended recipients of the email message by one or more RCPT command( One per recipient)
8. After all RCPT commands, the client sends a DATA command, to indicate that the client is ready to start transmission.
9. The server responds back with start mail input message which indicated that it is ready to accept the email message
10. The client sends the email message an identifier provided by the server to indicate that the transmission is over.

Some of the more common SMTP security provisions include:

- Checking the IP address of a device attempting connection and refusing to even start an SMTP session unless it is in a list of authorized client devices.
- Restriction of certain commands or features, such as e-mail relaying, to authorized users or client servers. This is sometimes done by requiring authentication via the SMTP extention *AUTH* before the command will be accepted.
- Limiting the use of commands such as *EXPN* to prevent unauthorized users from determining the e-mail addresses of users on mailing lists.
- Checking the validity of envelope information before accepting a message for delivery. Some servers will first verify that the originator's e-mail address is valid before agreeing to accept the *MAIL* command. Many will check the recipient's address and refuse the message if delivery is not to a local mailbox. Others use even more advanced techniques.
- Limiting the size of e-mail messages that may be sent or the number that may be sent in a given period of time.
- Logging all access to the server to keep records of server use and check for abuse.

## 1.2PGP :(Pretty Good Privacy):

### Introduction:

Phil Zimmerman introduced the Pretty Good Privacy (PGP) protocol.

### Characteristics:

1. It supports the basic requirements of cryptography.
2. It is quite simple to use and its source code and documentation is completely free to use.
3. It is more widely used comparing to PEM.
4. It mainly supports Encryption, Non repudiation and integrity of the message.

### How PGP works?

PGP combines some of the best features of both conventional and public key cryptography. PGP is a *hybrid cryptosystem.* When a user encrypts plaintext with PGP, PGP first compresses the plaintext. Data compression saves modem transmission time and disk space and, more importantly, strengthens cryptographic security. Most cryptanalysis techniques exploit patterns found in the plaintext to crack the cipher. Compression reduces these patterns in the plaintext, thereby greatly enhancing resistance to cryptanalysis. (Files that are too short to compress or which don't compress well aren't compressed.)

PGP then creates a *session key,* which is a one-time-only secret key. This key is a random number generated from the random movements of your mouse and the keystrokes you type. This session key works with a very secure, fast conventional encryption algorithm to encrypt the plaintext; the result is cipher text. Once the data is encrypted, the session key is then encrypted to the recipient's public key. This public key-encrypted session key is transmitted along with the cipher text to the recipient.

Decryption works in the reverse. The recipient's copy of PGP uses his or her private key to recover the temporary session key, which PGP then uses to decrypt the conventionally-encrypted cipher text.

The combination of the two encryption methods combines the convenience of public key encryption with the speed of conventional encryption. Conventional encryption is about 1, 000 times faster than public key encryption. Public key encryption in

turn provides a solution to key distribution and data transmission issues. Used together, performance and key distribution are improved without any sacrifice in security.

**Steps in PGP:**

1. Digital Signature
2. Compression
3. Encryption
4. Digital enveloping
5. Base – 64 encoding.

**Step1: Digital signatures:**

A digital signature serves the same purpose as a handwritten signature. However, a handwritten signature is easy to counterfeit. A digital signature is superior to a handwritten signature in that it is nearly impossible to counterfeit, plus it attests to the contents of the information as well as to the identity of the signer.
The basic manner in which digital signatures are, Instead of encrypting information using someone else's public key, user encrypts it with his own private key. If the information can be decrypted with user's public key, then it must have originated from the very same user.

**Step 2: Compression:**

An improvement on the above scheme is the addition of a one-way *hash function* in the process. A one-way hash function takes variable-length input — in this case, a message of any length, even thousands or millions of bits — and produces a fixed-length output; say, 160-bits. The hash function ensures that, if the information is changed in any way — even by just one bit — an entirely different output value is produced.

PGP uses a cryptographically strong hash function on the plaintext the user is signing. This generates a fixed-length data item known as a *message digest* Then PGP uses the digest and the private key to create the "signature." PGP transmits the signature and the plaintext together. Upon receipt of the message, the recipient uses PGP to recompute the digest, thus verifying the signature. PGP can encrypt the plaintext or not; signing plaintext is useful if some of the recipients are not interested in or capable of verifying the signature.

**Step 3: Encryption:**

The compressed output of step 2 is encrypted with a symmetric key. Normally IDEA algorithm in CFB mode is used.

**Step 4: Digital enveloping:**

The output of step 3 is now encrypted with the receiver's public key. The output of step 3 and step 4 together form a digital envelope.

**PGP certificates:**

A digital certificate is data that functions much like a physical certificate. A digital certificate is information included with a person's public key that helps others verify that a key is genuine or *valid.* Digital certificates are used to thwart attempts to substitute one person's key for another.

A digital certificate consists of three things:

- A public key.
- Certificate information. ("Identity" information about the user, such as name, user ID, and so on.)
- One or more digital signatures.

The purpose of the digital signature on a certificate is to state that the certificate information has been attested to by some other person or entity. The digital signature does not attest to the authenticity of the certificate as a whole; it vouches only that the signed identity information goes along with, or *is bound to,* the public key.

**Certificate formats**

A digital certificate is basically a collection of identifying information bound together with a public key and signed by a trusted third party to prove its authenticity. A digital certificate can be one of a number of different *formats.*

PGP recognizes two different certificate formats:

- PGP certificates
- X.509 certificates

**PGP certificate format**

A PGP certificate includes (but is not limited to) the following information:

- **The PGP version number —** this identifies which version of PGP was used to create the key associated with the certificate.

- **The certificate holder's public keys —** the public portion of your key pair, together with the algorithm of the key: RSA, DH (Diffie-Hellman), or DSA (Digital Signature Algorithm).
- **The certificate holder's information —** this consists of "identity" information about the user, such as his or her name, user ID, photograph, and so on.
- **The digital signature of the certificate owner —** also called a *self-signature;* this is the signature using the corresponding private key of the public key associated with the certificate.
- **The certificate's validity period —** the certificate's start date/ time and expiration date/ time; indicates when the certificate will expire.
- **The preferred symmetric encryption algorithm for the key —** indicates the encryption algorithm to which the certificate owner prefers to have information encrypted. The supported algorithms are CAST, IDEA or Triple-DES.

One unique aspect of the PGP certificate format is that a single certificate can contain multiple signatures. Several or many people may sign the key/ identification pair to attest to their own assurance that the public key definitely belongs to the specified owner.

**Trust models**

In relatively closed systems, such as within a small company, it is easy to trace a certification path back to the root. However, users must often communicate with people outside of their corporate environment, including some whom they have never met, such as vendors, customers, clients, associates, and so on. Establishing a line of trust to those who have not been explicitly trusted by your CA is difficult.

Companies follow one or another *trust model,* which dictates how users will go about establishing certificate validity. There are three different models:

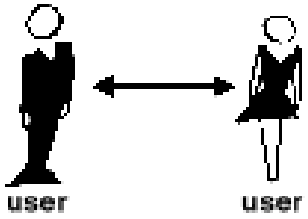
- Direct Trust
- Hierarchical Trust
- A Web of Trust

**Direct Trust:**

Direct trust is the simplest trust model. In this model, a user trusts that a key is valid because he or she knows where it came from. All cryptosystems use this form of trust in some way. For example, in web browsers, the root Certification Authority keys are directly trusted because they were shipped by the manufacturer. If

there is any form of hierarchy, it extends from these directly trusted certificates.

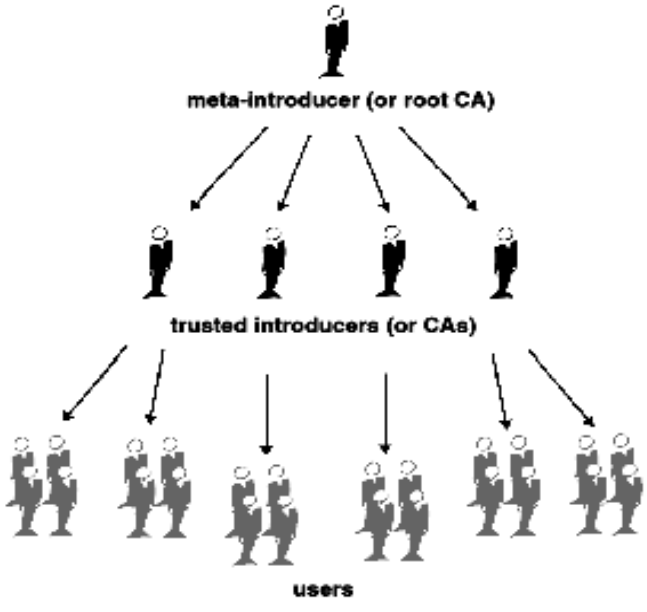
In PGP, a user who validates keys herself and never sets another certificate to be a trusted introducer is using direct trust.




**Direct trust**

**Hierarchical Trust:**

In a hierarchical system, there are a number of "root" certificates from which trust extends. These certificates may certify certificates themselves, or they may certify certificates that certify still other certificates down some chain. Consider it as a big trust "tree." The "leaf" certificate's validity is verified by tracing backward from its certifier, to other certifiers, until a directly trusted root certificate is found.




**Hierarchical trust**

**Web of Trust:**

A web of trust encompasses both of the other models, but also adds the notion that trust is in the eye of the beholder (which is the real-world view) and the idea that more information is better. It is thus a cumulative trust model. A certificate might be trusted directly, or trusted in some chain going back to a directly trusted

root certificate (the meta-introducer), or by some group of introducers.

Perhaps you've heard of the term *six degrees of separation,* which suggests that any person in the world can determine some link to any other person in the world using six or fewer other people as intermediaries. This is a web of introducers.

It is also the PGP view of trust. PGP uses digital signatures as its form of introduction. When any user signs another's key, he or she becomes an introducer of that key. As this process goes on, it establishes a *web of trust.*

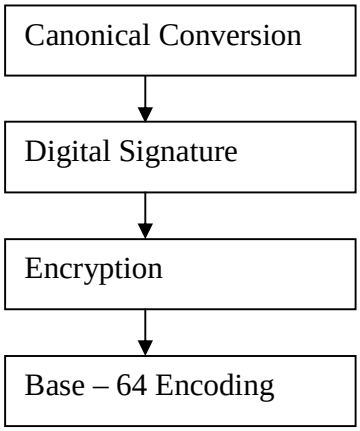
In a PGP environment, *any* user can act as a certifying authority. Any PGP user can validate another PGP user's public key certificate. However, such a certificate is only valid to another user if the relying party recognizes the validator as a trusted introducer.

**1.3 PEM (Privacy Enhanced Mail):**

**Introduction:**

1. This is an email security standard adopted by the Internet Architecture Board(IAB) to provide secure electronic mail communication over the Internet .
2. This was initially developed by the Internet Research Task Force(IRTF) and Privacy Security Force(IETF) PEM working Group.
3. PEM specifications are covered by RFC numbers 1421 to 1424.
4. This provides three main cryptographic functions: Encryption, Non-repudiation, Message integrity.

**1. PEM functions:**



**2. Canonical conversion:**

PEM transforms each email message into an abstract, canonical representation. This representation will allow the communication among the systems regardless of its architecture and the operating system of the sending and the receiving computers. This step is essential because in the internetworking system works on different architecture and operating system.

**3. Digital Signature:**

This works in a typical model by using any message digest algorithm such as MD2, MD5 etc. The message digest is then encrypted with the sender's private key to form the sender's digital signature.

**4. Encryption:**

The original email and the digital signature are encrypted together with a symmetric key.
This method uses DES or DES-3 algorithm in CBC mode.

**5. Base 64 encoding:**

This is the last step in PEM. This process transforms arbitrary binary input into printable character output. In this technique, the binary input is processed in blocks of 3 octets or 24 bits are considered to be made up of 4 sets, each of 6 bits. Each such set of 6 bits is mapped into an 8 – bit output character in this process.

## 2. IP Security

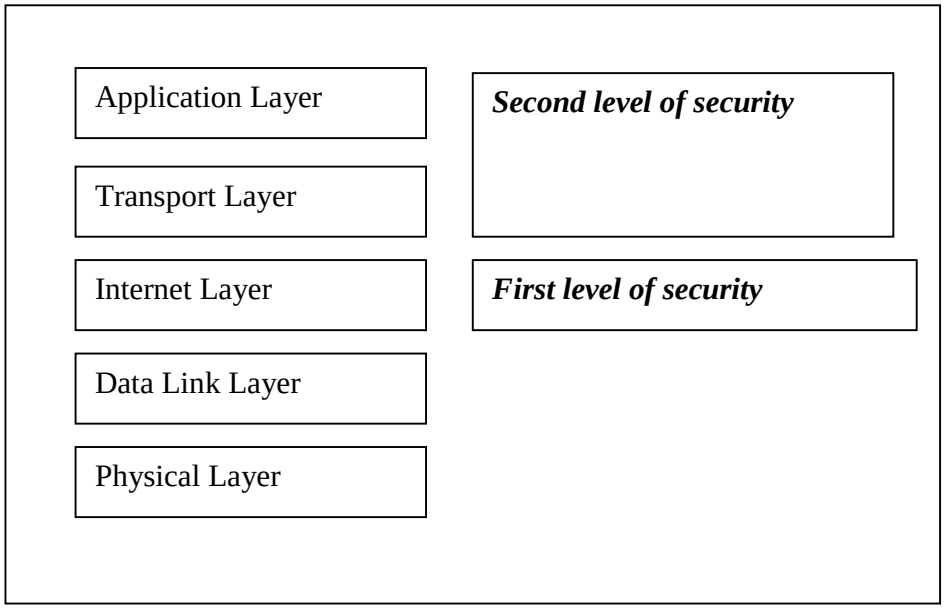
### 2.1 Introduction:

**Internet Protocol Security** (**IPsec**) is a protocol suite for securing Internet Protocol (IP) communications by authenticating and encrypting each IP packet of a data stream. This security mechanism is very important because IP packets are transffered in a plain text format.

Two levels of security in this scheme:

1. Offer security at the IP packet level itself.
2. Implementing the higher level security mechanisms, depending on the requirements.

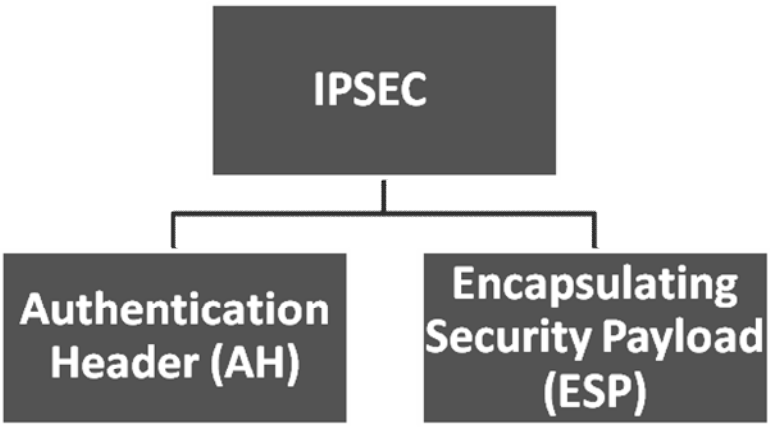
**Levels of Security:**




The Internet Architecture Board(IAB) prepared a report called as security in the Internet architectre(RFC1636). The outcome of the report and the study conducted by IAB , in 19995 IETF publiched five security based standards related to IPSec.Ipv4 supports all the standards mentioned by IETF.

## 2.2 Security architecture

- Internet Key Exchange (IKE and IKEv2) to set up a security association (SA) by handling negotiation of protocols and algorithms and to generate the encryption and authentication keys to be used by IPsec.
- Authentication Header(AH) to provide connectionless integrity and data origin authentication for Ipdatagrams and to provide protection against replay attacks.
- Encapsulating Security Payload (ESP) to provide confidentiality, data origin authentication, connectionless integrity, an anti-replay service (a form of partial sequence integrity), and limited traffic flow confidentiality.




**Authentication Header**

Authentication Header (AH) is a member of the IPsec protocol suite. It provides authentication, integrity and an optional anti-replay service. The IPSec AH is a header in an IP packet, which contains a cryptographic checksum for the contents of the packet. This is simply inserted between the IP header and any subsequent packet contents.

AH operates directly on top of IP, using IP protocol number 51.

**AH packet datagram:**

| **0 - 7 bit** | **8 - 15 bit** | **16 - 23 bit** | **24 - 31 bit** |
|---|---|---|---|
| Next header | Payload length | RESERVED | |
| Security parameters index (SPI) | | | |
| Sequence number | | | |
| Authentication data (variable) | | | |

**Meaning:**

Next header

The Next Header is an 8-bit field that identifies the type of the next payload after the Authentication Header. The value of this field is chosen from the set of IP Protocol Numbers defined in the most recent "Assigned Numbers.

Payload length

Size of AH packet.

RESERVED

Reserved for future use (all zero until then).

Security parameters index (SPI)

Identifies the security parameters, which, in combination with the IP address, then identify the security association implemented with this packet.

Sequence number

A monotonically increasing number, used to prevent replay attacks.

Authentication data

Contains the integrity check value (ICV) necessary to authenticate the packet; it may contain padding.

## 2.3 Encapsulating Security Payload

This provides data confidentiality. The ESP protocol also defines a new header to be inserted into the IP packet. ESP Processing also includes the transformation of the protected data into an unreadable, encrypted format. Normally the EXP will be inside the AH.

**ESP packet format**:

<table>
<tr><th>0 - 7 bit</th><th>8 - 15 bit</th><th>16 - 23 bit</th><th>24 - 31 bit</th></tr>
<tr><td colspan="4">Security parameters index (SPI)</td></tr>
<tr><td colspan="4">Sequence number</td></tr>
<tr><td colspan="4">Payload data (variable)</td></tr>
<tr><td></td><td colspan="2">Padding (0-255 bytes)</td><td></td></tr>
<tr><td></td><td></td><td>Pad Length</td><td>Next Header</td></tr>
<tr><td colspan="4">Authentication Data (variable)</td></tr>
</table>

**Field meanings:**

Security parameters index (SPI)

Identifies the security parameters in combination with IP address.

Sequence number

A monotonically increasing number, used to prevent replay attacks..

Payload data

The data to be transferred.

Padding

Used with some block ciphers to pad the data to the full length of a block.

Pad length

Size of padding in bytes.

Next header

Identifies the protocol of the payload data. The value of this field is chosen from the set of IP Protocol Numbers defined in the most recent "Assigned Numbers" .

Authentication data

Contains the data used to authenticate the packet.

## 2.4 Modes of operation

IPsec can be implemented in a host-to-host transport mode, as well as in a network tunnel **mode.**

### 1. Tunnel mode:

In tunnel mode, an encrypted tennel is established between two hosts. In this mode IPSec protects the entire IP datagram. It takes an IP datagram , adds the IPSec header and trailer and encrypts the whole thing. IT then adds new IP header to this encrypted datagram.

### 2. Transport mode

- In transport mode, only the payload (the data you transfer) of the IP packet is encrypted and/or authenticated. The routing is intact, since the IP header is neither modified nor encrypted; however, when the authentication header is used, the IP addresses cannot be translated, this will invalidate the hash value.. The transport and application layers are always secured by hash, so they cannot be modified in any way. Transport mode is used .

### 3. Network Management:

A network management system comprises:

- Network elements—Sometimes called managed devices, network elements are hardware devices such as computers, routers, and terminal servers that are connected to networks.

- Agents—Agents are software modules that reside in network elements. They collect and store management information such as the number of error packets received by a network element.

- Managed object—A managed object is a characteristic of something that can be managed. For example, a list of currently active TCP circuits in a particular host computer is a managed object. Managed objects differ from variables, which are particular object instances. Using our example, an object instance is a single active TCP circuit in a particular host computer. Managed objects can be scalar (defining a single object instance) or tabular (defining multiple, related instances).

- Management information base (MIB) -- A MIB is a collection of managed objects residing in a virtual information store. Collections of related managed objects are defined in specific MIB modules.

- Syntax notation—A syntax notation is a language used to describe a MIB's managed objects in a machine-independent format. Consistent use of a syntax notation allows different types of computers to share information. Internet management systems use a subset of the International Organization for Standardization's (ISO's) Open System Interconnection (OSI) Abstract Syntax Notation (ASN.1) to define both the packets exchanged by the management protocol and the objects that are to be managed.

- Structure of Management Information (SMI) -- The SMI defines the rules for describing management information. The SMI is defined using ASN.1.

- Network management stations (NMSs) -- Sometimes called consoles, these devices execute management applications that monitor and control network elements. Physically, NMSs are usually engineering workstation-caliber computers with fast CPUs, megapixel color displays, substantial memory, and abundant disk space. At least one NMS must be present in each managed environment.

- Parties—Newly defined in SNMPv2, a party is a logical SNMPv2 entity that can initiate or receive SNMPv2 communication. Each SNMPv2 party comprises a single, unique party identity, a logical network location, a single authentication protocol, and a single privacy protocol. SNMPv2 messages are communicated between two parties.

An SNMPv2 entity can define multiple parties, each with different parameters. For example, different parties can use different authentication and/or privacy protocols.

- Management protocol—A management protocol is used to convey management information between agents and NMSs. SNMP is the Internet community's de facto standard management protocol.

### 3.1 Overview

The Simple Network Management Protocol, also known as SNMP, is a vital protocol for Network Administrators. The SNMP Protocol uses IP with UDP or TCP. The SNMP protocol allows an Administrator to request information about one or more network devices hardware, software or configuration from a single point of management.

The SNMP Protocol has two sides, the agent and the management stations. The agent sends data about itself to the management station. The management station collects data from all the agents on the network. The protocol is not identical on each side. The agent sends alerts called traps and answers requests that were sent by the management station. The management station catches and decodes the traps. The management station also requests specific information from the agent.

The agent is a server, router, printer, bridge or workstations. Any network device that can use the IP protocol can be an SNMP agent. The software written to monitor the agents' hardware and send alerts is primarily the responsibility of the vendor. The initial design of the SNMP protocol called for every vendor to define the same hardware variables in their SNMP alerts. This was so any type of management station could decode the alerts. Today's technologically advance hardware is too different to follow this design. This is why vendors are responsible for SNMP agent software. This has also caused the development of SNMPv2 and SNMPv3.

The SNMP agent will monitor hardware and send a trap is any problems occur. For example, if a network printer runs low on toner or a server has a hard drive go bad, a trap will be sent to the management station. The management station catches the trap and acts on the trap as configured by the administrator. The administrator could configure the management station to send an e-mail or have the administrator paged.

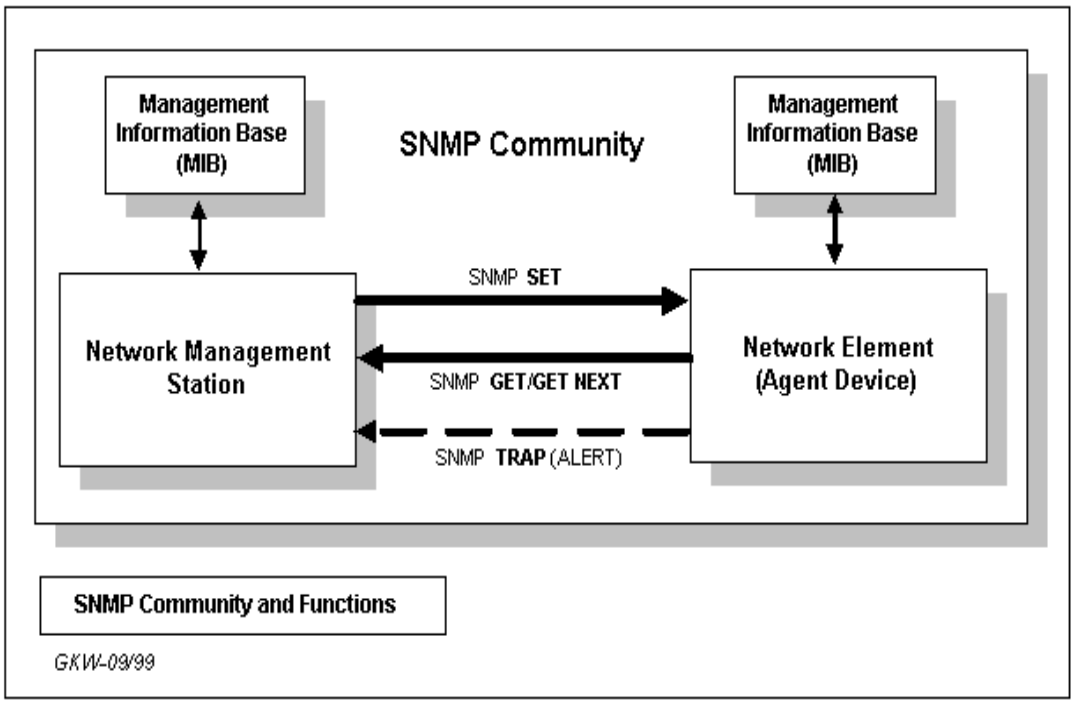
The management station may also send a request to the agent. In this case the SNMP agent will reply to the management station with the specific data. For example the management station may wish to view the agents hardware event log and see the current BIOS level. The agent will reply with SNMP packets containing the hardware's events and BIOS level. The information is passed through requests and replies with the use of the MIB or the Management Information Base.

The management station is responsible for decoding the SNMP packets and providing an interface to the administrator. The interface can be a GUI or command line. The interface is the key to the SNMP protocol. The single point of management for administering several network devices.

### 3.2 Management Information Base (MIB)

SNMP lets TCP/IP-based network management clients use a TCP/IP-based internetwork to exchange information about the configuration and status of nodes. The information available is defined by a set of **managed objects** referred to as the SNMP Management Information Base (MIB). The subset of managed objects that make up the TCP/IP portion of the MIB is maintained by each TCP/IP node. SNMP can also generates trap messages used to report significant TCP/IP events asynchronously to interested clients.

SNMP applications run in a network management station (NMS) and issue queries to gather information about the status, configuration, and performance of external network devices (called *network elements* in SNMP terminology). For example, HP Open view software is an example of a network management station, and a Cisco 4500 Router with its SNMP agent enabled could be considered an example of a network element. The following picture is indicating the simple architecture.




SNMP agents run in network elements and respond to NMS queries (GETs). In addition, agents send unsolicited reports (called *traps*) back to the NMS when certain network activity occurs. These traps can spawn events such as e-mail alerts, automatic pages or network server parameter modifications.

For security reasons, the SNMP agent validates each request from an SNMP manager before responding to the request, by verifying that the manager belongs to an SNMP *community* with access privileges to the agent.

An SNMP community is a logical relationship between an SNMP agent and one or more SNMP managers. The community has a name, and all members of a community have the same access privileges: either read-only (members can view configuration and performance information) or read-write (members can view configuration and performance information, and also change the configuration).

All SNMP message exchanges consist of a community name and a data field, which contains the SNMP operation and its associated operands. You can configure the SNMP agent to receive requests and send responses only from managers that are members of a known community. If the agent knows the community name in the SNMP message and knows that the manager generating the request is a member of that community, it considers the message to be authentic and gives it the access allowed for members of that community. Thus, the SNMP

community prevents unauthorized managers from viewing or changing the configuration of a router or other SNMP manageable device.

**In summary, the SNMP Management program performs the following operations:**

- The **GET** operation receives a specific value about a managed object, such as available hard disk space from the agent's MIB.
- The **GET-NEXT** operation returns the "next" value by traversing the MIB database (tree) of managed object variables.
- The **SET** operation changes the value of a managed object's variable. Only variables whose object definition allows READ/WRITE access can be changed.
- The **TRAP** operation sends a message to the Management Station when a change occurs in a managed object (and that change is deemed important enough to spawn an alert message).

### 3.3 Security

Security is important when using SNMP. Because SNMP agents broadcasts information and in some agents changed, security must not be overlooked. The initial version of SNMP now referred to, as SNMPv1 did not have a very good implementation of security. SNMP faces all the standard threats of any network application: Modification of Information, Masquerade, Message Stream Modification, and Disclosure.

SNMPv1 used only one form of security, community names. Community names are similar to passwords. Agents can be set to reply to queries only received by accepted community names. In SNMPv1 the community name was passed along with the data packet in clear text. This allowed anyone to ease drop and learn the SNMP community name or password.

SNMPv2 brought a lot of extra security. First of all everything in the packet except for the destination address is encrypted. Inside the encrypted data is the community name and source IP address. The agent can now decode the encrypted data packet and use the accepted community name and accepted source IP address to validate the request. This type of security is referred to as party and context. Party referring to a specific machine or person and context referring to a name or string associated with the party. SNMP uses DES (Data Encryption Standard) for encrypting the data packets.

SNMPv3 provides the latest architecture for SNMP security. It incorporates an SNMP context engine ID to encode and decode SNMP contexts. The context engine ID could take more time than allowed to explain. In short it matches a context name with an object and the security requires the object and context to match. SNMPv3 provides three levels of security. The highest level is with authentication and privacy. The middle level is with authentication and no privacy and the bottom level is without authentication or privacy.

- The perfect example of why SNMP security is important is its ability to reboot devices. Administrators cannot let that ability be violated. The latest version of SNMP have brought security a long way from clear text.

**Questions:**

1. Write a short note on Network management security .
2. Define the following
    a) SMTP
    b) PGP
    c) PEM
3. Write in deail about the Email security mechnism
4. Explain the need for Mangemnt Information Base
5. Write in detail about SNMP

# 9

# SECURITY FOR ELECTRONIC COMMERCE

1. SECURE ELECTRONIC TRANSACTION (SET)
    1. Introduction
    2. Sample SET session
    3. Objectives
    4. SET participants
    5. SET process

2. SECURE SOCKET LAYER (SSL)
    1. Introduction
    2. Protocol stack
    3. Position of SSL in TCP/IP
    4. Three sub protocols
        1) Handshake Protocol
        2) Alert Protocol
        3) Record Protocol

## 1. SET protocol:

### 1.1Introduction:

Secure Electronic Transaction(SET) is an open encryption and security specification that is designed for prtoecting credit card transactions on the Internet. The initial work in this area was started by Mastercard and Visa in the year 1996. In 1998 with the jont ventrue of companies like IBM, Microsoft ect the first generation of SET compliant products appeared in the market. This protocl is providing a tramendous help towards the ecommerce business transaction.

**Basic services:**

1. It provides a secure communication channel among all the parties involved in an e-commerce transaction.
2. It provides authentication by the use of digital certificates .
3. It ensures confidentiality, because the information is only available to the parties involved in a transaction and that tooo only when and where necessary.

## 1.2 A Sample SET Session

Before getting into details of SET, we shall take a simple example to describe how SET works from the consumer's perspective.

1. The consumer accesses the merchant's web site, goes through the various goods on display and selects what he or she wants. Perhaps there is a virtual shopping cart where he or she drops all the items to be purchased. At the end, the customer proceeds to the virtual checkout counter. A screen pops up giving details, including the cost of all the items the shopper is purchasing, plus taxes and shipping costs.
2. Then the screen asks for the payment method and the consumer chooses to pay through a credit card using SET.
3. Immediately, special software on the consumer's PC called Digital Wallet is invoked, and it asks the customer to choose one credit card from the many he or she possesses.
4. The consumer chooses a card, and the electronic transaction using SET is underway. A few seconds later, there is a confirmation that this order has been processed.

## 1.3 Objectives of SET

SET addresses seven major business requirements

1. Provide confidentiality of payment information and enable confidentiality of order information that is transmitted along with the payment information.
2. Ensure the integrity of all transmitted data.
3. Provide authentication that a cardholder is a legitimate user of a branded payment card account.
4. Provide authentication that a merchant can accept branded payment card transactions through its relationship with an acquiring financial institution.

5. Ensure the use of the best security practices and system design techniques to protect all legitimate parties in an electronic commerce transaction.
6. Create a protocol that neither depends on transport security mechanisms nor prevents their use.
7. Facilitate and encourage interoperability among software and network providers.

## 1.4 SET participants:

There are mainly six entities involved in SET. The next section briefly discusses these entities.

### 1. Certificate Authority (CA)

The Certificate Authority (CA) issues and manages digital certificates that provide proof of identity of all parties involved in a SET transaction. It manages the creation and distribution of digital certificates for cardholders, merchants and payment gateways.

### 2. SET Payment Gateway

The payment gateway is the bridge between SET and the existing payment network. The payment gateway translates SET messages for the existing payment system to complete an electronic transaction.

### 3. SET Merchant Point of Sale Server

A merchant offers goods or services for sale in the Internet and accepts electronic credit card payments. Merchant that accepts payment cards must have a relationship with an acquirer. The merchant Point of Sale Server provides an interface between the cardholder and the acquirer payment gateway.

### 4. Cardholder and Electronic Wallet

Cardholder is an authorized holder of a payment card supported and issued by an issuing bank. Cardholders use electronic wallets to store digital representations of credit cards and make purchases with them. SET ensures that the interactions the cardholder has with a merchant keep the payment card account information confidential.

### 5. Acquiring Bank

An acquirer is the financial institution that establishes an account with a merchant and processes payment card authorizations and payments.

**6. Issuing Bank**

The issuing bank establishes an account for a cardholder and issues the payment card to the cardholder. The issuer guarantees payment for authorized transactions using the payment card.

***Card holders:***

Card holder is the authorized holder of the card that has been issued by an Issuer.

***Merchant:***

A merchant is a person or a organization that wants to sell goods or services to cardholders.

***Issuers:***

The issuer is a financial institution that provides a payment card to a cardholder.

***Acquirer:***

This is a financial institution that has a relationship with merchants for processing payment card authorizations and payments. The acquirer also provides electronic funds transfer to the merchant account.

***Payment Gateway:***

This can be taken up by the acquirer or an organization as a dedicated function. This processes the payment messages on behalf of the merchant. The payment gateway acts as an interface between SET and the existing card payment networks for payment authorization.

**1.5 SET process:**

- The customer opens a MasterCard or Visa bank account. Any issuer of a credit card is some kind of bank.
- The customer receives a digital certificate.. This electronic file functions as a credit card for online purchases or other transactions. It includes a public key with an expiration date. It has been through a digital switch to the bank to ensure its validity.
- Third-party merchants also receive certificates from the bank. These certificates include the merchant's public key and the bank's public key.
- The customer places an order over a Web page, by phone, or some other means.

- The customer's browser receives and confirms from the merchant's certificate that the merchant is valid.
- The browser sends the order information. This message is encrypted with the merchant's public key, the payment information, which is encrypted with the bank's public key (which can't be read by the merchant), and information that ensures the payment can only be used with this particular order.
- The merchant verifies the customer by checking the digital signature on the customer's certificate. This may be done by referring the certificate to the bank or to a third-party verifier.
- The merchant sends the order message along to the bank. This includes the bank's public key, the customer's payment information (which the merchant can't decode), and the merchant's certificate.
- The bank verifies the merchant and the message. The bank uses the digital signature on the certificate with the message and verifies the payment part of the message.
- The bank digitally signs and sends authorization to the merchant, who can then fill the order.

## 2. SSL Protocol (Secure Socket Layer):

### 2.1 INTRODUCTION:

This is an internet protocol for secure exchange of information between a web browser and a web server. It provides a secure pipe between the web browser and the web server. This protocol was developed by Netscape Corporation developed in 1994. The SSL is supporting two basic security services:

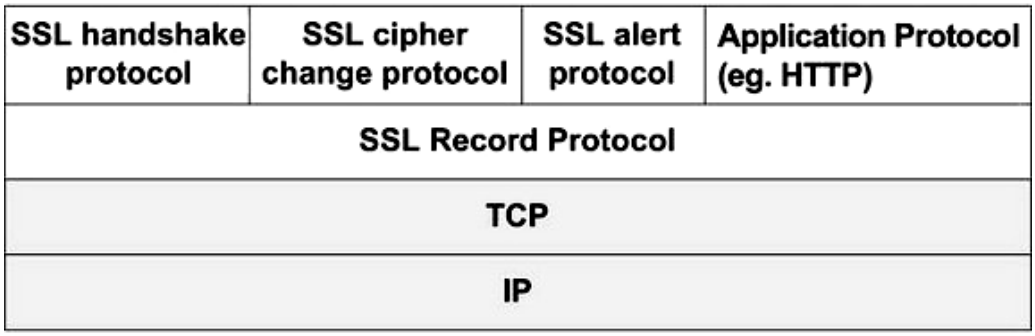
1. Authentication
2. Confidentiality

- Authenticating the client and server to each other: the SSL protocol supports the use of standard key cryptographic techniques (public key encryption) to authenticate the communicating parties to each other. Though the most frequent application consists in authenticating the service client on the basis of a certificate, SSL may also use the same methods to authenticate the client.
- Ensuring data integrity: during a session, data cannot be either intentionally or unintentionally tampered with.
- Securing data privacy: data in transport between the client and the server must be protected from interception and be

readable only by the intended recipient. This prerequisite is necessary for both the data associated with the protocol itself (securing traffic during negotiations) and the application data that is sent during the session itself. SSL is in fact not a single protocol but rather a set of protocols that can additionally be further divided in two layers:

1. the protocol to ensure data security and integrity: this layer is composed of the SSL Record Protocol,
2. the protocols that are designed to establish an SSL connection: three protocols are used in this layer: the SSL Handshake Protocol, the SSL ChangeCipher SpecPprotocol and the SSL Alert Protocol.

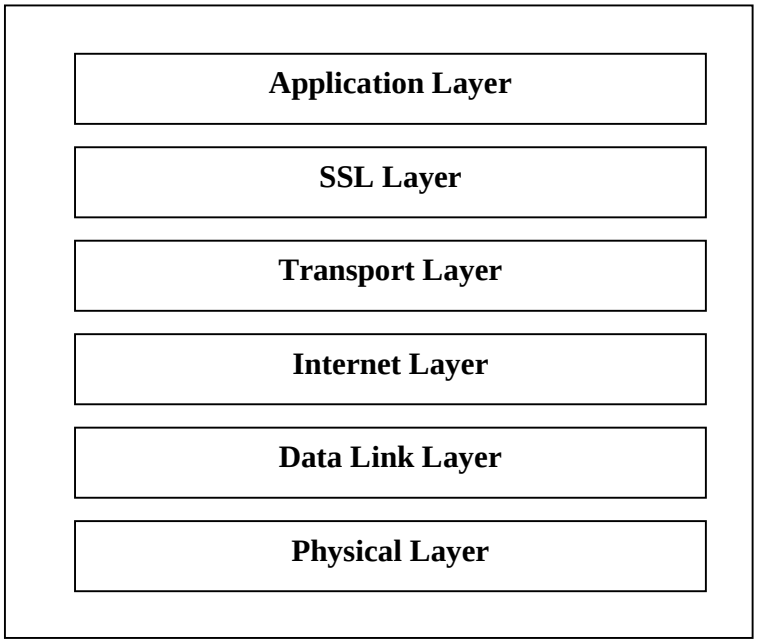
**2.2The SSL protocol stack :**




**The SSL protocol stack:**

SSL uses these protocols to address the tasks as described above. The SSL record protocol is responsible for data encryption and integrity. As can be seen in Figure 2, it is also used to encapsulate data sent by other SSL protocols, and therefore, it is also involved in the tasks associated with the SSL check data. The other three protocols cover the areas of session management, cryptographic parameter management and transfer of SSL messages between the client and the server. Prior to going into a more detailed discussion of the role of individual protocols and their functions let us describe two fundamental concepts related to the use of SSL.

## 2.3 Position of SSL in TCP/IP:




## 2.4 Three sub-protocols:

1. Handshake protocol
2. Record Protocol
3. Alert Protocol

### 2.4.1.Handshake protocol:

The handshake protocol constitutes the most complex part of the SSL protocol. It is used to initiate a session between the server and the client. Within the message of this protocol, various components such as algorithms and keys used for data encryption are negotiated. Due to this protocol, it is possible to authenticate the parties to each other and negotiate appropriate parameters of the session between them.

The process of negotiations between the client and the server is illustrated in Figure 4. It can be divided into 4 phases separated with horizontal broken lines. During the first phase, a logical connection must be initiated between the client and the server followed by the negotiation on the connection parameters. The client sends the server a client_hello message containing data such as:

- Version: The highest SSL version supported by the client,
- Random: data consisting of a 32-bit timestamp and 28 bytes of randomly generated data. This data is used to protect the

key exchange session between the parties of the connection.

- Session ID: a number that defines the session identifier. A nonzero value of this field indicates that the client wishes to update the parameters of an existing connection or establish a new connection on this session. A zero value in this field indicates that the client wishes to establish a new connection.
- CipherSuite: a list of encryption algorithms and key exchange method supported by the client. The server, in response to the client_hello message sends a server_hello message, containing the same set of fields as the client message, placing the following data:
- Version: the lowest version number of the SSL protocol supported by the server,
- random data: the same fashion as used by the client, but the data generated is completely independent,
- session ID: if the client field was nonzero, the same value is sent back; otherwise the server's session ID field contains the value for a new session,
- CipherSuite : the server uses this field to send a single set of protocols selected by the server from those proposed by the client. The first element of this field is a chosen method of exchange of cryptographic keys between the client and the server. The next element is the specification of encryption algorithms and hash functions, which will be used within the session being initiated, along with all specific parameters.

The set of encryption algorithms and key exchange method sent in the CipherSuite field establishes three components:

- the method of key exchange between the server and client,
- the encryption algorithm for data encryption purposes,
- a function used for obtaining the MAC value.

The server begins the next phase of negotiations by sending its certificate to the client for authentication. The message sent to the client contains one or a chain of X509 certificates. These are necessary for authentication of both the server and the certification path towards a trusted certification official of the certificating body for the server. This step is not obligatory and may be omitted, if the negotiated method of key exchange does not require sending the certificate (in the case of anonymous Diffie-Hellman method). Depending on the negotiated method of key exchange, the server may send an additional server_key_exchange message, which is

however not required in the case when the fixed Diffie-Hellman method or RSA key exchange technique has been negotiated. Moreover, the server can request a certificate from the client. The final step of Phase 2 is the server_done message, which has no parameters and is sent by the server merely to indicate the end of the server messages. After sending this message, the server waits for a client response. Upon receipt of the message, the client should verify the server's certificate, the certificate validation data and path, as well as any other parameters sent by the server in the server_hello message. The client's verification consists of:

- Validation date check of the certificate and comparison with the current date, to verify whether the certificate is still valid,
- checking whether the certifying body is included in the list of trusted Certifying Authorities in possession of the client. If the CA, which has issued the server's certificate is not included in the CAs list, the client attempts to verify the CA signature. If no information about the CA can be obtained, the client terminates the identification procedure by either returning the error signal or signalling the problem for the user to solve it.
- Identifying the authenticity of the public key of the CA which has issued the certificate: if the Certifying Authority is included in the client's list of trusted CAs, the client checks the CA's public key stated in the server's certificate with the public key available from the list. This procedure verifies the authenticity of the certifying body.
- checking whether the domain name used in the certificate matches the server name shown in the server's certificate.

Upon successful completion of all steps the server is considered authenticated. If all parameters are matched and the server's certificate correctly verified, the client sends the server one or multiple messages. Next is the client__key_exchange message, which must be sent to deliver the keys. The content of this message depends on the negotiated method of key exchange. Moreover, at the server's request, the client's certificate is sent along with the message enabling verification of the certificate. This procedure ends Phase 3 of negotiations.

Phase 4 is to confirm the messages so far received and to verify whether the pending data is correct. The client sends a change_cipher_spec message (in accordance with the pending SSL ChangeCipher Spec), and then sets up the pending set of algorithm parameters and keys into the current set of the same. Then the client sends the finished message, which is first protected with just negotiated algorithms, keys and secrets. This is to confirm

that the negotiated parameters and data are correct. The server in response to the client sends the same message sequence. If the finished message is correctly read by either party this confirms that the transmitted data, negotiated algorithms and the session key are correct. This indicates that the session has been terminated and that it is possible to send the application data between the server and the client, via SSL. At this point the TCP session between the client and the server is closed, however a session state is maintained, allowing it to resume communications within the session using the retained parameters.

It is worth noticing that both Phases 2 and 3 are used by both parties to verify the authenticity of the server's certificate and possibly the client's certificate during the handshake step. If the server cannot be successfully authenticated by the client on the basis of the delivered certificate, the handshake terminates and the client will generate an error message. The same will occur at the server if the client's certificate authenticity cannot be confirmed.

### 2.4.2 The Record protocol:

This protocol evolves after the successful completion of the handshake protocol. This protocol provides two services to an SSL connection.

1. Confidentiality
2. Integrity

The SSL record protocol involves using SSL in a secure manner and with message integrity ensured. To this end it is used by upper layer SSL protocols. The purpose of the SSL record protocol is to take an application message to be transmitted, fragment the data which needs to be sent, encapsulate it with appropriate headers and create an object just called a record, which is encrypted and can be forwarded for sending under the TCP protocol. The first step in the preparation of transmission of the application data consists in its fragmentation i.e. breaking up the data stream to be transmitted into 16Kb (or smaller) data fragments followed by the process of their conversion in a record. These data fragments may be further compressed, although the SSL 3.0 protocol specification includes no compression protocol, thus at present, no data compression is used.
At this moment, creation of the record is started for each data portion by adding a header to it, possible information to complete the required data size and the MAC. The record header that is added to each data portion contains two elementary pieces of information, namely the length of the record and the length of the data block added to the original data.
In the next step, the record data constructed consists of the following elements:

- primary data,
- some padding to complete the datagram as required,
- MAC value.

MAC is responsible for the verification of integrity of the message included in the transmitted record. It is the result of a hash function that follows a specific hash algorithm, for example MD5 or SHA-1. MAC is determined as a result of a hash function that receives the following data:

MAC = Hash function [secret key, primary data, padding, sequence number].

A secret key in creation of MAC is either a client write MAC secret or a server write MAC secret respectively, it depends on which party prepares the packet. After receiving the packet, the receiving party computes its own value of the MAC and compares it with that received. If the two values match, this means that data has not been modified during the transmission over the network. The length of the MAC obtained in this way depends on the method uses for its computing. Next, the data plus the MAC are encrypted using a preset symmetric encryption algorithm, for example DES or triple DES. Both data and MAC are encrypted. This prepared data is attached with the following header fields:

- Content type: identifies what payload is delivered by the packet to determine which higher protocols are to be used for processing of data included in the packet. The possible values are change_cipher_spec, alert, handshake, and application_data that refer to the appropriate protocols.
- Major version : establishes the main portion of the protocol version to be used. For SSL 3.0, the value is 3,
- Minor version : establishes the additional portion of the used version of the protocol. For SSL 3.0 the value is 0.

With the addition of fields, the process of record preparation is completed. Afterwards, the record is sent to the targeted point. The entire process of preparation of the packet to be sent is illustrated in Figure 3.

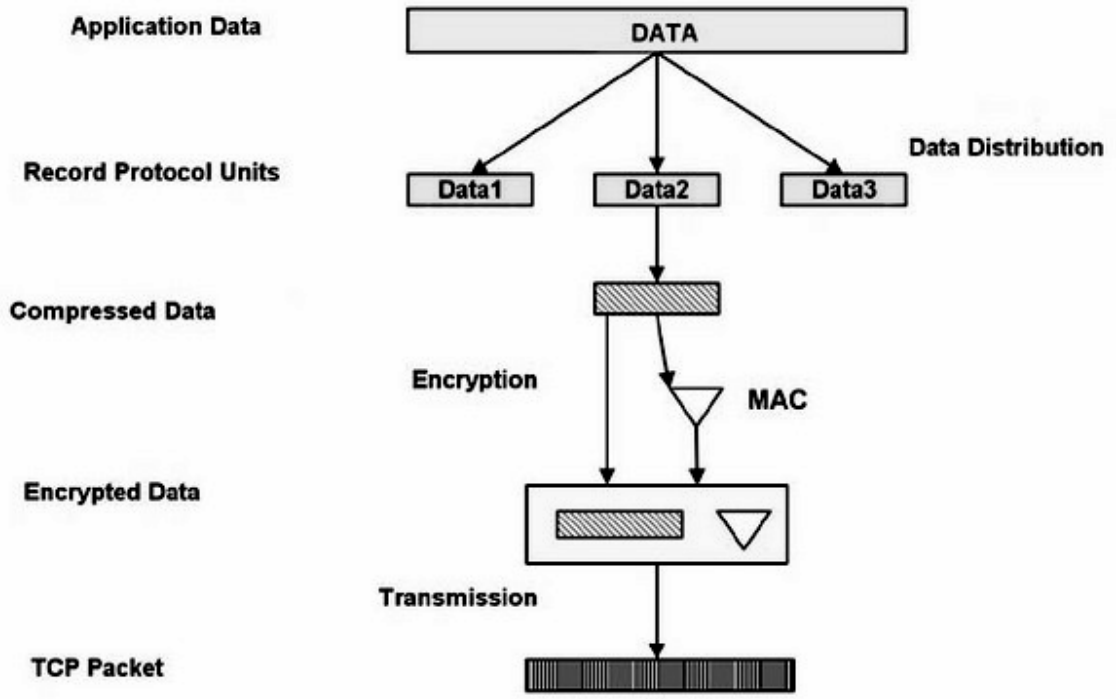



***Figure 3. Creating a packet under SSL record protocol***

**2.4.3 Alert protocol:**

The Alert Protocol is used by parties to convey session messages associated with data exchange and functioning of the protocol. Each message in the alert protocol consists of two bytes. The first byte always takes a value, “warning” (1) or “fatal” (2) , that determines the severity of the message sent. Sending a message having a „fatal” status by either party will result in an immediate termination of the SSL session. The next byte of the message contains one of the defined error codes, which may occur during an SSL communication session.

**QUESTIONS:**

1. Explain the objectives of SET protocol
2. Write in detail about the process involved in SET protocol
3. Give a sample SET session and explain each step involved in detail
4. Wrtie a short note on SSL protocol
5. Define the following
    a) Alert protocol
    b) Record protocol
    c) Handshake protocol

❖❖❖❖

# 10

# SYSTEM SECURITY

1. Intruders and Virus
    - Virus development
    - Components
    - Structure
    - Types
2. Firewalls
    - Introduction
    - Need
    - Protection
    - Limitations
3. Intrusion Detection
    - Introduction
    - Types

## 1. Intruders:

Intruders are the people who get the access to the system or the files by using various following methods by which create harm to the system and the programs.

## Social Engineering:

Social Engineering is hacker – speak for tricking a person into revealing some vital information. In other words, Social Engineering is a practice of cheating people into revealing sensitive data on a computer system, often on the Internet.

This is like an art, a special tool of the attacker in which he plays the psychological tricks at the target in order to gain the important information. All this happens without the knowledge of the target, i.e. the target does not know at all that he/she is giving some vital information to the hacker.

There are various methods, which are used by hackers for social engineering. Some of which are described below:

1. ***Using personal conversation*:** By talking directly with the target, the hacker tries to grab some important information. The hacker may have a friendly chat with the user and pretend like a friendly person and grab the details by getting the confidence of the user. To achieve the same the hacker will start the conversation by himself giving his own details. As the conversation is initiated by the user himself, this is more dangerous since the hacker may be revealing himself in the process. This needs the communication skill and convincing capability to the hacker. But this is the easiest method among all. This technique is often used to trick a business into disclosing customer information, and is used by private investigators to obtain telephone records, utility records, banking records and other information directly from junior company service representatives.

2. ***Through Telephonic conversation:*** Using this, the hacker may pose as a top official, network administrator & may persuade the target in giving vital information about them. As the hacker is staying away from the user and using telephone as a medium to precede the conversation, this method is not revealing the hacker directly and is the safer method. This needs the capability of taking the conversation further with the name of the legitimate user. This is somewhat easier than first method and is much widely used.

3. ***By chatting with the target:*** In this kind of method, the hacker keeping himself secret, can chat & make good friendship with the target & then using trickier questions can gain the required information. This is the well known and widely used method among the various users.

4. ***By sending anonymous mails:*** It is possible to send mails without needing a sender's address or by using some address, which looks official and legitimate enough to fool the target in giving key information. This is also widely used technique.

**Bugs and Backdoors:**

A bug may refer to some kind of problem in the software, which is undesired by its author. It may mean some sort of limitation in the software, which does not allow it to do the appropriate work. These are the loopholes or vulnerabilities in the program, which make it less secure.

Hackers, who know about these loopholes, can misuse it or use it for their own benefit whereas some of them may disclose it to make everyone aware about it.

One solution for this is to keep the software always updated with bug fixes, which are normally provided by its developer.

For example, added virus databases in an antivirus utility, service packs for operating systems etc. One should use them regularly in order to stay away from the new viruses or vulnerabilities.

A bug is something in a program that does not meet its specification. They are thus particularly hard to model because, by definition, which assumptions if any will fail. The effect of a bug is not necessarily limited to ill effects or abuses of the particular service involved. Rather, the entire system can be penetrated because of one failed component. There is no perfect defense, but there are steps one can take to shift the odds.

- The administrator should be checking for all the input correctness at every point.
- If the program has fixed size buffers of any sort, then it should be made sure that they do not overflow.
- If we use dynamic memory allocation, prepare for memory or file system exhaustion, and proper recovery strategies, which may need memory or disk space, too.

**Backdoors:**

Another type of security vulnerability is due to the backdoors *(also called as trapdoors).*

A backdoor is a feature of a program that can be used to make it act in some way that the person who is running it did not intend. Among IRC-related programs, bots, clients and scripts can have backdoors. An important point to note is that some backdoors are intentional and some are not. It is easy for someone who doesn't really understand what he's doing to put unwanted backdoors. In clients and bots, which are usually written in C, the bugs or backdoors tend to be harder to find and exploit. The line between an unintended backdoor and a bug are rather thin. A backdoor if it can be used to make the bot do something specific, and just a bug if it can be used only to make the client or bot disconnect (ping timeout, or excess flood).

Another type of security vulnerability is due to the backdoors *(also called as trapdoors).* These are the programs which when

stored on the target systems, may allow easy access to hackers or give them sufficient information about the target to carry out the attacks. There are several backdoor programs used by the hackers. These are like automated tools, which carry out the destructive jobs for the hackers. Trojan horse programs may also come into this category. In order to save from the backdoors, cleaner solutions are also available (which work in similar manner as the antivirus utilities).

- Back doors are shortcut entry points to software or networks i.e. entry without going through authentication mechanisms.
- Designers for convenience in error detection while compiling might create them and hackers taking advantage of bugs or weaknesses in the program might create inadvertently left unclosed or such backdoors.
- The only solution for backdoor attacks is double and triple checking of every piece of software before implementation.

A back door is a feature of a program that can be used to make it act in some way that the person who is running it, did not intend to do it.

**Authentication Failures:**

This error message indicates that the authentication process between your local computer and the remote host computer has for some reason failed. The most common cause for failed authentication is an incorrect password, likely caused by a typing mistake.

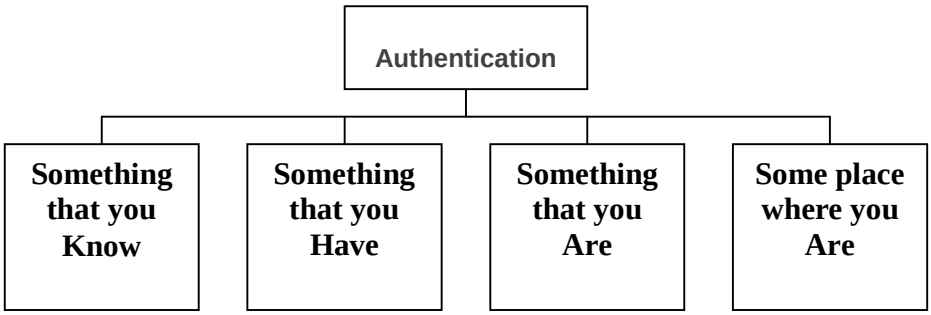
Also the user name may be incorrect. So it is necessary to check that the typing has been done correctly.

One possible reason for authentication failure is that the remote host computer may have been configured to require several authentication methods to be used. For example both password and public key authentication could be used for increased security. Even if the password is typed incorrectly, some other required authentication method could have failed. A relatively common situation is one where the remote host computer is expecting public-key authentication and the user has not sent the public key to the host. It may also be possible that the user account on the remote host computer has been disabled or that the remote host computer is having temporary problems causing errors with the login procedure.

Try to connect again and carefully type in your user name and password. If after a couple of retries you are sure that you have

entered both of them correctly, contact the system administrator of the remote host computer

Authentication can be stated as the method of validating the identity of genuine or authorized users.




**Something that you know:**

- The very first and the foremost are your user-id and passwords.
- The next can be your personal matters such as your date of birth, your mother's maiden name, your pet's name etc.

These are simple to use and require no special hardware; user-id and password continue to be the most popular method of authentication.

**Something that you have:**

- Image of person's face
- Retina or iris
- Fingerprints
- Hand geometry
- Digital Signature

**Something that you are:**

This is the actual physical feature of the user like the fingerprints or the way the user communicates (Voice) or the way that the user looks (Image) etc. These are the natural qualities of the user, which cannot be changed or misused even by the user himself.

There are various methods used for this purpose, but the most commonly used one is by way of login name and passwords. In order to keep your authentication method foolproof, some strict policies have to be adopted. But, still the authentication failure is one of the ways in which the intruders can penetrate into the systems.

Firstly the passwords have to be properly designed using all the available rules. Sometimes, if the password is stored in some user database in clear text, then the intruder can easily intercept it another example of authentication failure is by way of a fake login program run on a terminal.

One more form of authentication attack may come from the remote login programs. Protocols like rlogin, telnet is vulnerable to this. If these are available on for your host, intruders may keep retrying till they are lucky and get a chance to penetrate these systems. Hence, normally it is advised to turn often-remote login features for added security.

Most of the attacks that take place are as a result of some authentication failure. But authentication failures or authentication race refers to the tactic of beating a one-time password scheme that works with many security systems.

Usually a one-time password is a good technique of ensuring that the password even if intercepted and understood will not have any significance since its not going to be used again. But even then eavesdroppers can easily pick up a plain password on an unencrypted session and they may take a shot at single time passwords also.

For this we assume an example of a password that contains only digits and is of known length. The attacker initiates ten connections to the desired service. Each connection is waiting for the same unknown password. The valid user connects and starts typing the correct password.

The attack program watches this, and relays the correct characters to its ten connections as they arc typed. When anyone digit remains to be entered, the program sends a different digit to each of its connections, before the valid user can type the last digit. Because the computer is faster, it wins the race, and one of the connections is validated. These authentication schemes often allow only a single login with each password, so the valid user will be rejected, and will have to try again. Of course, in this case the attacker needs to know the length of the password.

**1. VIRUS**

Viruses have restricted propagating mechanisms and are parasitic in nature (need other host programs to propagate). Most of them carry a payload that is the action(s) they perform after infection.

There is nothing magical about computer viruses. A virus is simply a computer program, which is stored somewhere on the disk. But unlike the other programs, this program is not available separately. It will try to hide itself by attaching to some legitimate program. Still, it is to be called a computer program. There is very much similarity between a computer virus and a biological virus. Both invade the body/machine and attach to cell / program and once lodged, they monitor the activity of the host in order to replicate themselves.

### 1.1 Why viruses are developed?

The virus programs are normally written with an objective of destructive effects. A virus cannot do anything that was not written into its program. It is the creation of intelligent computer programmer but with the harmful intentions. There are other harmful programs like worms, Trojan horses etc. A program is not a virus unless it has the ability to replicate itself.

There are several ways and intentions with which the viruses are written. These range from complete destruction of host system, to simple pass-time nuisance activities. Sometimes viruses are used to stop from copying legitimate programs, or just to prove someone's knowledge, to simply make fun out of it & so forth.

### 1.2 Overview:

When an infected file (any file that has the virus attached) is executed, the virus also gets executed, thereby infecting that system. It does this by making copies of itself and attaching or injecting them into other files available .The impact varies from low to high levels. Most viruses typically destroy specific file types, either by deleting the contents of the file or by encrypting the contents with a random key and corrupting the boot sectors / metadata areas / file system tables (FAT tables in Windows / inode metadata in Linux).

Certain viruses merely executes certain instruction sets which in turn could enable or disable certain functionality; slow down all the processes by consuming CPU cycles and even memory.

Another category of viruses could disable the existing defense mechanisms such as Antivirus software or firewall thereby permitting other malicious programs to infect the system.

## 1.3 Components of Virus:

There are 3 parts:

a. Infector
b. Replicator
c. Payload

**a. Infector:**

This is the section of the viral code, which infects some part of the system when triggered for the first time. That happens when the virus enters a system first. The infector part may decide to infect a file system or a system sector or an application.

**b. Replicator:**

The Replicator is that section of the virus, which has the Job of making the virus replicate or duplicate such that every time the viral code is triggered or executed, the virus gets a chance to replicate. Usually the infection at a proper place like the system sector leaves a better chance of replication more number of times. Replicators are crucial in deciding the strength of the virus overall.

**c. Payload:**

This the section, which can determine the amount of damage or harm the virus can cause to the system or its resources. Usually it is a direct implication of the Replicator. Because explicator will be able to replicate a specific number of times, so the payload will be greater. Depending on how much the payload is, the virus goes into the attack phase. Usually a higher payload means that the virus can easily overcome the system and its resources can be easily overcome by the virus.

## 1.4 Structure of Viruses

A virus can be pre-pended or post-pended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.

Program V: =

{ goto main ;
1234567;
subroutine infect- executable : =
{

```
 loop;
file : = get- random- executable-file:
if(first-line-of-file=1234567)
then goto loop
else prepend V to file;
}
subroutine do-damage :=
{ whatever damage is to be done}

subroutine trigger-pulled :=
{Return true if some condition holds}

Main: main-program: =
{Infect executable;
if trigger pulled then  do   damage;
goto next;}
next:
}
```

In this case, the virus code, V, is prepended to infected programs, and it is assumed that the entry point to the program, when invoked is the first line of the program.

An infected program begins with the virus code and works as follows. The first line of code is a jump to the main virus program. The second line is a special marker that is used by the virus to determine whether or not a potential victim program has already been infected with this virus. When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them.

Next, the virus may perform some action, usually detrimental to the system. This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions. Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.

Here is a simple structure of a virus. In the infected binary, at a known byte location in the title, a virus inserts a signature byte used to determine if a potential carrier program has been previously infected.

```
V()
{
 infectExecutable( );
 if(triggered ( ))
 {
   doDamage( );
 }
    jump to main of infected program;
}

Void infectExecutable ( )
{
file = chose an uninfected executable file;
 prepend V to file/void doDamage( )
{
   int triggered( )
    {
          return (some test? 1: 0);
    }
}
}
```

The above virus makes the infected file longer than it was, making it easy to spot. There are many techniques to leave the tile length and even a check sum unchanged and yet infect. For example, many executable tiles often contain long sequences of zero bytes, which can be replaced by the virus and re-generated. It is also possible to compress the original executable code like the typical Zip programs do, and uncompress before execution and pad with bytes so that the check sum comes out to be what it was.

A virus such as the one just described is easily detected because an infected version of a program is longer than the corresponding uninfected one. A way to thwart such a simple means of detecting a virus is to compress the executable file so that both the infected and uninfected versions are of identical length.

### 1.4.1 Logic for a Compression Virus

Program CV: =

{goto main;

01234567;

Subroutine infect-executable: =

{Loop:

File: = get-random-executable-file;

if(first-line-of-file = 01234567) then goto loop;

(1) Compress file;

(2) prepend CV to file;

}

Main: main-program: =

{If ask-permission then infect-executable;

(3) Uncompress rest-of-file;

(4) Run uncompressed file;}

}

**Life Stages of a virus:**

## 1.4.2 Life Stages of a Virus

There arc 3 stages to a Virus Life Cycle:

a. Infection Phase
b. Replication Phase.
c. Attack Phase

## 1.4.3 Infection Phase:

When the virus executes it has the potential to infect other programs. What's often not clearly understood is precisely when it will infect the other programs. Some viruses infect other programs each time they are executed; other viruses infect only upon a certain trigger. This trigger could be anything; a day or time, an external event on your PC, a counter within the virus, etc. Virus writers want their programs to spread as far as possible before anyone notices them.

You can never be sure the virus simply hasn't yet triggered its infection phase.

Many viruses go resident in the memory of your PC in the same or similar way as terminate and stay resident (TSR) programs. TSRs are programs that executed under DOS but stayed in memory instead of ending. This means the virus can wait for some external event before it infects additional programs. The virus may silently lurk in memory waiting for you to access a diskette, copy a file, or execute a progran1, before it infects anything. This makes viruses more difficult to analyze since it's hard to guess what trigger condition they use for their infection.

Resident viruses frequently take over portions of the system software on the PC to hide their existence. This technique is called

stealth. Polymorphic techniques also help viruses to infect yet avoid detection.

### 1.4.4 Replication Phase:

This is the phase where the virus replicates or duplicates or infects more system files, so that its strength is increased enough to attack the system and over come the system's protective resources.

Replication is an important phase, because with proper replication, the attack phase of the virus can become that much more lethal. Hence, replication is planned around some trigger, which the viral code needs to trigger its duplicate. In order to do this, virus is placed in an executable file or the system sector. Usually in an executable file, the virus will be able to replicate only if the executable file is executed. Hence, the system sector is a good bet, since every time if the system boots and the boot sector is where the viral code is placed, then the virus gets a chance to replicate every time.

### 1.4.5 Attack Phase:

Many viruses do unpleasant things such as deleting files or changing random data on your disk, simulating typos or merely slowing your PC down. Just as the infection phase can be triggered by some event, the attack phase also has its own trigger.

Viruses often delay revealing their presence by launching their attack only after they have had ample opportunity to spread. This means the attack could be delayed fur days, weeks, months, or even years after the initial infection.

Usually, viruses can be also triggered to attack under the arrival of a specific time. In that case, the virus constantly checks whether the pre-decided time of attack has arrived or not. If it has, then the virus unleashes the attack phase and starts doing what it is meant to do.

## 1.5. SUBTYPES and WORKING

Classification of viruses can be done as follows:

1. Memory Based (How they live (stay) in memory)
2. Target Based (How they spread to others)
3. Obfuscation Technique Based (What they do to hide)
4. Payload Based (What they do after infection)


Fig-02: A Virus Model

Fig-02: A Virus Model

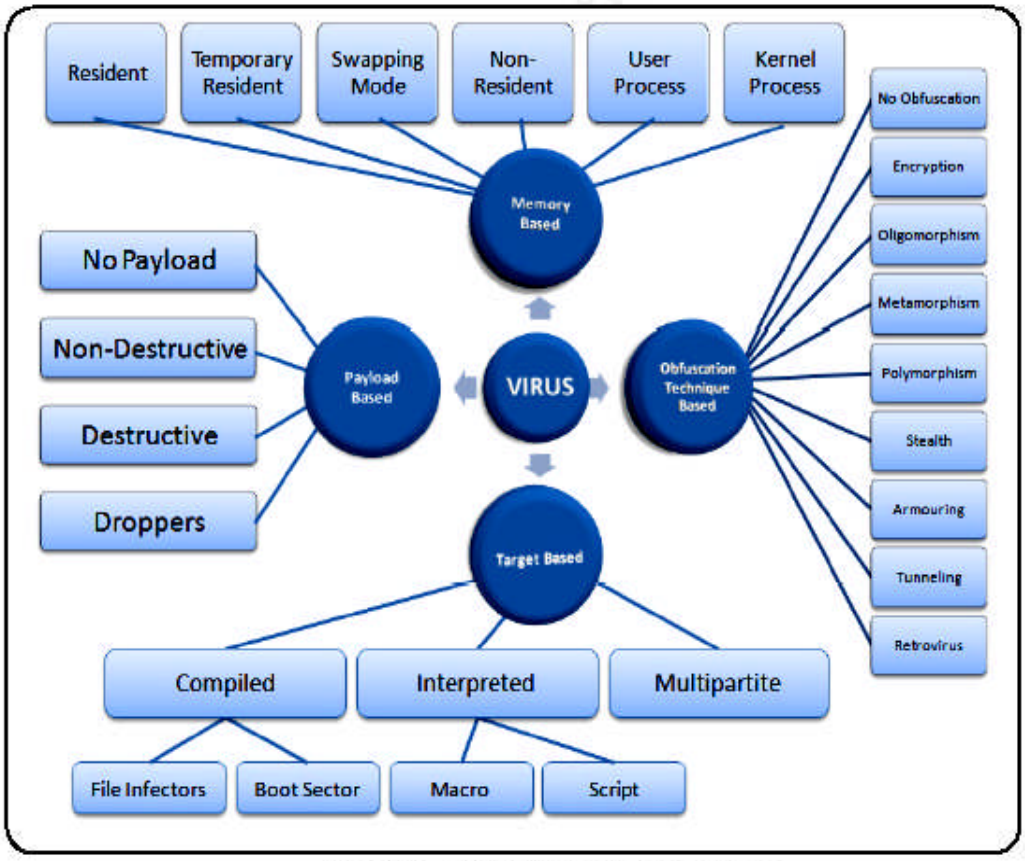



Fig-03: VIRUS Classification

### 1.5.1. Memory Based classification:

One method of classifying viruses is based on the way they operate in the memory. There are six subtypes according to this classification (Szor, 2005, chap. 5), namely,

1. Resident (In memory)
2. Temporary Resident (In memory temporarily)
3. Swapping Mode (Only a part loaded in memory temporarily)
4. Non-Resident (Not in memory)
5. User Process (As a user level process)
6. Kernel Process (As a process in the kernel)

### 1.5.2. Target based classification

1. Appending Virus
2. Prepending Virus
3. Overwriting Virus
4. Cavity Virus
5. Compressing Virus
6. Amoeba Virus
7. EPO Virus
8. Companion Virus
9. Code Virus

**Viruses can be divided into nine (9) Subtypes,**

1. No Obfuscation
2. Encryption
3. Oligomorphism
4. Polymorphism
5. Metamorphism
6. Stealth
7. Armoring
8. Tunneling
9. Retro

## 2. Firewall:

### 2.1. Introduction:

### What are firewalls?

The Internet is a vital and growing network that is changing the way many organizations and individuals communicate and do business. Using the Internet we can get connected to any other computer, no matter how far the two are located from each other on the network. However, the Internet suffers from significant and widespread security problems. Many agencies and organizations have been attacked or probed by intruders, with resultant high losses to productivity and reputation. In some cases, organizations have had to disconnect from the Internet temporarily, and have invested significant resources in correcting problems with system and network configuration. Sites that are unaware of or ignorant of these problems face a significant risk that network intruders will attack them. Even sites that do observe good security practices face problems with new vulnerabilities in networking software and the persistence of some intruders. But this facility usually may be a nightmare for network support staff, which is left with a very difficult job of trying to protect the corporate networks from a variety of attacks. At a broad level, there are two kinds of attacks:

1. Most corporations have large amounts of valuable and confidential data in their networks. Leaking of this critical information to competitors can be a great setback.
2. Apart from the danger of the insider information leaking out, there is a great danger of the outside elements (such as viruses and Worms) entering a corporate network to create havoc.

- Some of the problems with Internet security are a result of inherent vulnerabilities in the services (and the protocols that the services implement), while others are a result of host configuration and access controls that are poorly implemented or overly complex to administer.
- Additionally, the role and importance of system management is often short-changed in job descriptions, resulting in many administrators being, at best, part-time and poorly prepared.

We will talk about these problems in detail.

## 2.3 Need of firewall

The Internet, like any other society, is plagued with the kind of jerks who enjoy the electronic equivalent of writing on other people's walls with spray paint, tearing their mailboxes off, or just sitting in the street blowing their car horns. Some people try to get real work done over the Internet, and others have sensitive or proprietary data they must protect. Usually, a firewall's purpose is to keep the jerks out of your network while still letting you get your job done.

Many traditional-style corporations and data centers have computing security policies and practices that must be adhered to. In a case where a company's policies dictate how data must be protected, a firewall is very important, since it is the embodiment of the corporate policy. Frequently, the hardest part of hooking to the Internet, if you're a large company, is not justifying the expense or effort, but convincing management that it's safe to do so. A firewall provides not only real security--it often plays an important role as a security blanket for management.

Lastly, a firewall can act as your **corporate ``ambassador''** to the Internet. Many corporations use their firewall systems as a place to store public information about corporate products and services, files to download, bug fixes, and so forth. Several of these systems have become important parts of the Internet service structure(e.g.: UUnet.uu.net, whitehouse.gov, gatekeeper.dec.com) and have reflected well on their organizational sponsors.

**The following list summarizes the primary benefits of using a firewall.**

- ✓ Protection from Vulnerable Services
- ✓ Controlled Access to Site Systems
- ✓ Concentrated Security
- ✓ Enhanced Privacy
- ✓ Logging and Statistics on Network Use, Misuse
- ✓ Policy Enforcement

## 2.4. What can a firewall protect against?

- Some firewalls permit only **email traffic** through them, thereby protecting the network against any attacks other than attacks against the email service. Other firewalls provide less strict protections, and block services that are known to be problems.

- Generally, firewalls are configured to protect against unauthenticated interactive logins from the ``outside'' world. This, more than anything, helps prevent vandals from logging into machines on your network. More elaborate firewalls block traffic from the outside to the inside, but permit users on the inside to communicate freely with the outside. The firewall can protect you against any type of network-borne attack if you unplug it.
- Firewalls are also important since they can provide a **single ``choke point''** where security and audit can be imposed. Unlike in a situation where someone dialing in with a modem is attacking a computer system, the firewall can act as an effective ``phone tap'' and tracing tool. Firewalls provide an important logging and auditing function; often they provide summaries to the administrator about what kinds and amount of traffic passed through it, how many attempts there were to break into it, etc.
- This is an important point: providing this ``choke point'' can serve the same purpose on your network as a guarded gate can for your site's physical premises. That means anytime you have a change in ``zones'' or levels of sensitivity, such a checkpoint is appropriate. A company rarely has only an outside gate and no receptionist or security staff to check badges on the way in. If there are layers of security on your site, it's reasonable to expect layers of security on your network.

## 2.5. Advantages of the Firewall:

- A firewall prevents unauthorized Internet users from accessing a private network connected to the Internet.
- It enforces a security policy by allowing a single point for implementing and controlling all security decisions to be made.
- It filters, monitors, and logs the sessions between any two networks. As a result, your exposure to the Internet is also limited.

## 2.6 Limitation of the Firewall:

The main limitations of a firewall can be listed as follows:

- **Insider's intrusion**: A firewall system is designed to thwart outside attacks. Therefore, if an inside user attacks the internal network in some way, the firewall cannot prevent such an attack.

- **Direct Internet traffic**: A firewall must be configured very carefully. It is effective only if it is the only-entry point of an organization's network. If, instead, the firewall is one of the entry-exit points, a user can bypass the firewall and exchange information with the Internet via the other entry exit points. This can open up the possibilities of attacks on the internal network through those points. The firewall cannot, obviously be expected to take care of such situations.

- **Virus attacks**: A firewall cannot protect the internal network from virus threats. This is because a firewall cannot be expected to scan every incoming file or packet for possible virus contents. Therefore, a separate virus detection and removal mechanism is required for preventing virus attacks. Alternatively, some vendors bundle their firewall products with anti virus software, to enable both the features out of the box.

- It needs **specialized skills** to configure, and many attacks occur because of badly configured policies on a firewall.

## 3. Intrusion Detection System (IDS):

### 3.1 Introduction:

An intrusion detection system (IDS) monitors network traffic and monitors for suspicious activity and alerts the system or network administrator. In some cases the IDS may also respond to anomalous or malicious traffic by taking action such as blocking the user or source IP address from accessing the network.

IDS come in a variety of “flavors” and approach the goal of detecting suspicious traffic in different ways. There are network based (NIDS) and host based (HIDS) intrusion detection systems. There are IDS that detect based on looking for specific signatures of known threats- similar to the way antivirus software typically detects and protects against malware- and there are IDS that detect based on comparing traffic patterns against a baseline and looking for anomalies. There are IDS that simply monitor and alert and there are IDS that perform an action or actions in response to a detected threat. We’ll cover each of these briefly.

### 3.2 TYPES:

**NIDS**

Network Intrusion Detection Systems are placed at a strategic point or points within the network to monitor traffic to and from all devices on the network. Ideally you would scan all inbound

and outbound traffic, however doing so might create a bottleneck that would impair the overall speed of the network.

**HIDS**

Host Intrusion Detection Systems are run on individual hosts or devices on the network. A HIDS monitors the inbound and outbound packets from the device only and will alert the user or administrator of suspicious activity is detected

**Signature Based**

A signature based IDS will monitor packets on the network and compare them against a database of signatures or attributes from known malicious threats. This is similar to the way most antivirus software detects malware. The issue is that there will be a lag between a new threat being discovered in the wild and the signature for detecting that threat being applied to your IDS. During that lag time your IDS would be unable to detect the new threat.

**Anomaly Based**

An IDS which is anomaly based will monitor network traffic and compare it against an established baseline. The baseline will identify what is “normal” for that network- what sort of bandwidth is generally used, what protocols are used, what ports and devices generally connect to each other- and alert the administrator or user when traffic is detected which is anomalous, or significantly different, than the baseline.

**Passive IDS**

A passive IDS simply detects and alerts. When suspicious or malicious traffic is detected an alert is generated and sent to the administrator or user and it is up to them to take action to block the activity or respond in some way.

**Reactive IDS**

Reactive IDS will not only detect suspicious or malicious traffic and alert the administrator, but will take pre-defined proactive actions to respond to the threat. Typically this means blocking any further network traffic from the source IP address or user.

One of the most well known and widely used intrusion detection systems is the open source, freely available snort It is available for a number of platforms and operating systems including both Linux and Windows. Snort has a large and loyal following and there are many resources available on the Internet where you can acquire signatures to implement to detect the latest threats.

There is a fine line between a firewall and an IDS. There is also technology called IPS – Intrusion Prevention System. An IPS is essentially a firewall which combines network-level and application-level filtering with a reactive IDS to proactively protect the network. It seems that as time goes on firewalls, IDS and IPS take on more attributes from each other and blur the line even more.

Essentially, your firewall is your first line of perimeter defense. Best practices recommend that your firewall be explicitly configured to DENY all incoming traffic and then you open up holes where necessary. You may need to open up port 80 to host web sites or port 21 to host an FTP file server. Each of these holes may be necessary from one standpoint, but they also represent possible vectors for malicious traffic to enter your network rather than being blocked by the firewall.

That is where your IDS would come in. Whether you implement a NIDS across the entire network or a HIDS on your specific device, the IDS will monitor the inbound and outbound traffic and identify suspicious or malicious traffic which may have somehow bypassed your firewall or it could possibly be originating from inside your network as well.

IDS can be a great tool for proactively monitoring and protecting your network from malicious activity, however they are also prone to false alarms. With just about any IDS solution you implement you will need to "tune it" once it is first installed. You need the IDS to be properly configured to recognize what is normal traffic on your network vs. what might be malicious traffic and you, or the administrators responsible for responding to IDS alerts, need to understand what the alerts mean and how to effectively respond.

**QUESTIONS:**

1. Explain the role of intruders
2. Discuss about the structure of virus
3. What are the various types of virus
4. Why do we need firewall
5. Explain the advantages of firewalls
6. What are the limitations of firewalls
7. Why do we need intrusion detection system
8. What are the various types of IDS
9. What are the various intrusion techniques