UNIT - I

1. Introduction: The need for security-security approaches-principles of security-Plain Text and Cipher Text-substitution and Transposition Techniques-Encryption and Decryption-Symmetric and Asymmetric Cryptography-Stenography-key range and key size-types of attacks.

2. Number Theory: Introduction to number theory- Modular Arithmetic, Euclidean algorithm, Euler theorem, Fermat Theorem, Totient Function, Multiplicative and Additive Inverse.

1. Introduction

Computer data often travels from one computer to another, leaving the safety of its protected physical surroundings. Once the data is out of hand, people with bad intention could modify or forge your data, either for amusement or for their own benefit.

Cryptography can reformat and transform our data, making it safer on its trip between computers. The technology is based on the essentials of secret codes, augmented by modern mathematics that protects our data in powerful ways.

1. **Computer Security:** Generic name for the collection of tools designed to protect data and to prevent hackers.
2. **Network Security:** Measures to protect data during their transmission.
3. **Internet Security:** Measures to protect data during their transmission over a collection of interconnected networks.

Security Attacks, Services and Mechanisms:

To assess the security needs of an organization effectively, the manager responsible for security needs some systematic way of defining the requirements for security and characterization of approaches to satisfy those requirements. One approach is to consider three aspects of information security:

1. **Security Attack:** Any action that compromises the security of information owned by an organization.
2. **Security Mechanism:** A mechanism that is designed to detect, prevent or recover from a security attack.
3. **Security Service:** A service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter

security attacks and they make use of one or more security mechanisms to provide the service.

Basic Concepts:

1. **Cryptography:** The art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form.
2. **Plaintext:** The original intelligible message.
3. **Cipher Text:** The transformed message.
4. **Cipher:** An algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods.
5. **Key:** Some critical information used by the cipher, known only to the sender& receiver.
6. **Encipher** (encode): The process of converting plaintext to cipher text using a cipher and a key.
7. **Decipher** (decode): The process of converting cipher text back into plaintext using a cipher and a key.
8. **Cryptanalysis:** The study of principles and methods of transforming an unintelligible message back into an intelligible message *without* knowledge of the key. This is also called **code breaking.**
9. **Cryptology:** Both cryptography and cryptanalysis.
10. **Code:** An algorithm for transforming an intelligible message into an unintelligible one using a code-book.

Cryptography:

Cryptographic systems are generally classified along 3 independent dimensions:

1. **Type of operations used for transforming plain text to cipher text:** All the encryption algorithms are based on two general principles: **substitution**, in which each element in the plaintext is mapped into another element, and **transposition**, in which elements in the plaintext are rearranged.
2. **The number of keys used:** If the sender and receiver uses same key then it is said to be **symmetric key (or) single key (or) conventional encryption**: If the sender and receiver use different keys then it is said to be **public key encryption.**
3. **The way in which the plain text is processed:** A **block cipher** processes the input and block of elements at a time, producing output block for each input block. A

stream cipher processes the input elements continuously, producing output element one at a time, as it goes along.

Cryptanalysis:

The process of attempting to discover X (Plain Text) or K (Key) or both is known as cryptanalysis. The strategy used by the cryptanalysis depends on the nature of the encryption scheme and the information available to the cryptanalyst.

There are various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst.

1. **Cipher text only:** A copy of cipher text alone is known to the cryptanalyst.
2. **Known plaintext:** The cryptanalyst has a copy of the cipher text and the corresponding plaintext.
3. **Chosen plaintext:** The cryptanalysts gains temporary access to the encryption machine. They cannot open it to find the key, however; they can encrypt a large number of suitably chosen plaintexts and try to use the resulting cipher texts to deduce the key.
4. **Chosen cipher text:** The cryptanalyst obtains temporary access to the decryption machine, uses it to decrypt several string of symbols, and tries to use the results to deduce the key.

Steganography:

A plaintext message may be hidden in any one of the two ways. The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text.

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

Ex: 1 The sequence of first letters of each word of the overall message spells out the real (Hidden) message.

2. Subset of the words of the overall message is used to convey the hidden message.

Various other techniques have been used historically, some of them are:

1. **Character marking:** Selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held to an angle to bright light.

2. **Invisible ink:** A number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.
3. **Pin punctures:** Small pin punctures on selected letters are ordinarily not visible unless the paper is held in front of the light. Typewritten correction ribbon – used between the lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Drawbacks of Steganography:

1. Requires a lot of overhead to hide a relatively few bits of information.
2. Once the system is discovered, it becomes virtually worthless.

Security Services (OR) Principles of Security:

The classification of security services are as follows:

1. **Confidentiality:** This ensures that only authorized parties can able to view the message on the system or message to be transmitted.
2. **Authentication:** This ensures that the origin of the message transmission is correctly identified by receiver.
3. **Integrity:** This ensures that only the authorized parties can able to modify the message which is transmitted and also ensures whether total message transmission is done.
4. **Non-repudiation:** Either sender or receiver can't be able to avoid the transmission of the message.
5. **Access control:** The authorized parties control the reading or writing of the particular file. It can be always done by the authorization parties.
6. **Availability:** The message in the system or the message that is to be transmitted is always available to the authorized parties sender or receiver.

Security Mechanisms:

One of the most specific security mechanisms in use is cryptographic techniques. Encryption or encryption-like transformations of information are the most common means of providing security. Some of the mechanisms are

1. **Encipherment:** The process of converting plaintext to cipher text using a cipher and a key.
2. **Digital Signature:** A digital code (generated and authenticated by public key encryption) which is attached to an electronically transmitted document to verify its contents and the sender's identity.

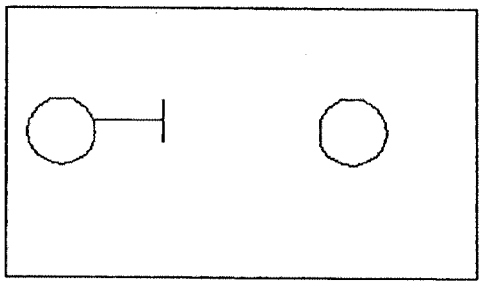
3. **Access Control:** The authorized parties control the reading or writing of the particular file. It can be always done by the authorization parties.

Security Attacks:

An attack on the security of a computer system may be defined as "Threat". There are four general categories of attack which are listed below.

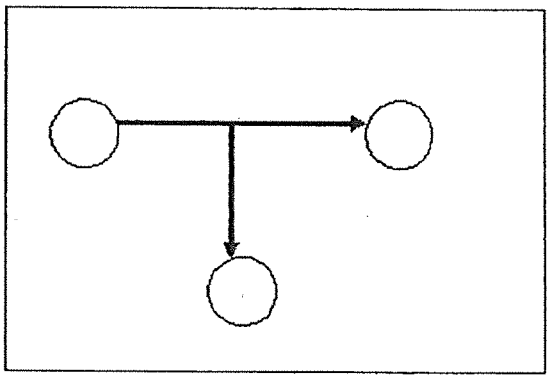
1. Interruption:

Here the message is destroyed in the middle and is made unavailable to the receiver. This is also called as an "***Attack on Availability***".



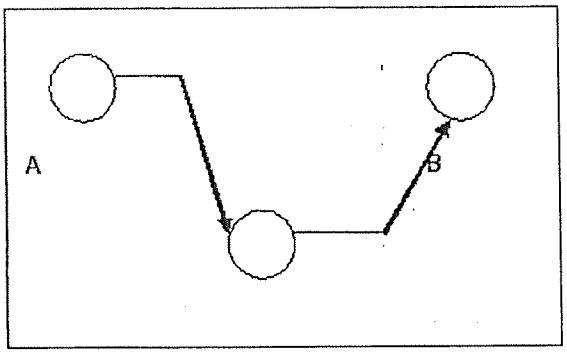
2. Interception:

Here the message is accessed by an unauthorized user during the process of the transmission. Here the receiver receives the message and not unaware of the introducers. This is called attack on "***Confidentiality***".



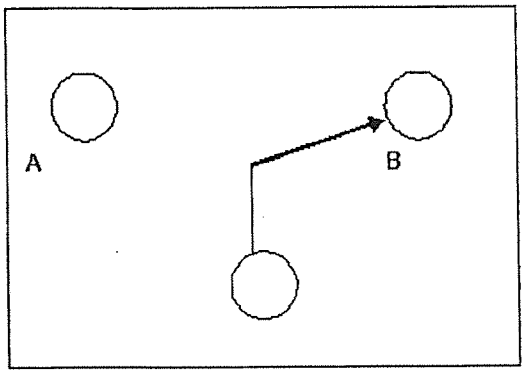
3. Modification:

Here the intruders or unauthorized party gains the access over the communication channel and retrieves the message from the sender and then modifies it, and send it to receiver. This is called as an "***Attack on Integrity***".



4. Fabrication:

Here an intruder will insert a message into the communication channel and send it to the receiver. This is also called as an "***Attack on Authentication***".



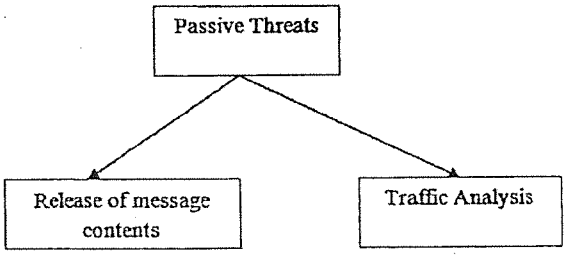
Cryptographic Attacks (OR) Types of Attacks:

An attack on computer systems security is defined as threat or attack. ***In a technologist's point of view***, the attacks are classified into two categories. They are

1. Passive attacks or Passive threats
2. Active attacks or Active threats

1. Passive attacks or Passive threats:

The passive threats are in the nature of monitoring and dropping. The goal of passive threats is to know the message contents which are being transmitted. Here the message is not altered in the middle. There are two types of passive attacks.



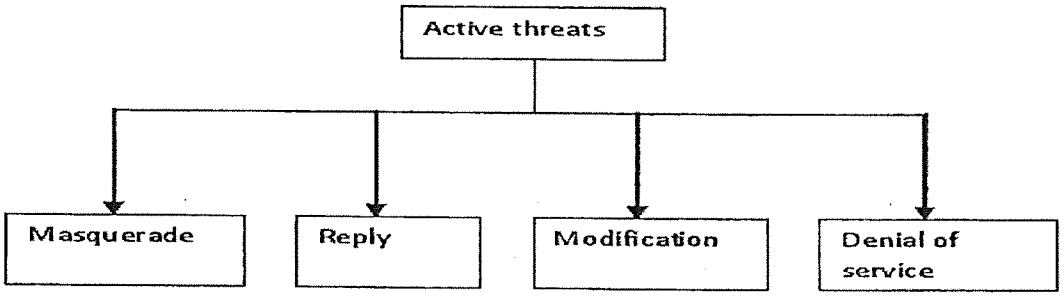
a. Release of message contents: Here the unauthorized user will know the message transmission.

Ex: A telephonic conversation or content of e-mail message etc.,

b. Traffic analysis: In the traffic analysis the unauthorized person always monitors the authorized person, analysis the no. of communications between them and also the length of the message transmission. He also analysis how many times the receiver going to respond and passing on these factors the unauthorized person guesses the message transmission.

2. Active attacks or Active threats:

In the active threats the message is altered in the middle. These threats can be classified as follows.



a. **Masquerade:** Here an unauthorized party or an intruder pretends to act like an authorized user and sends the message to all the other authorized users.
b. **Reply:** Here an unauthorized person copies the previous transmission between the authorized parties and send it to receiver no. of times as reply.
c. **Modification:** Here the message is being altered by the unauthorized party in the process of transmission and send to the receiver.
d. **Denial of service:** It prevents the normal use communication facilities like description of an entire network etc.,

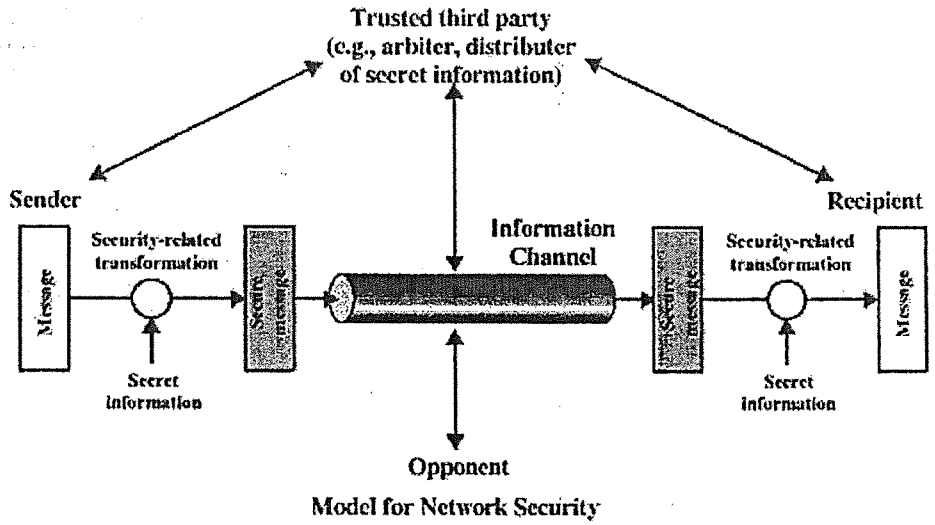
Symmetric and public key algorithms:

Encryption/Decryption methods fall into two categories.

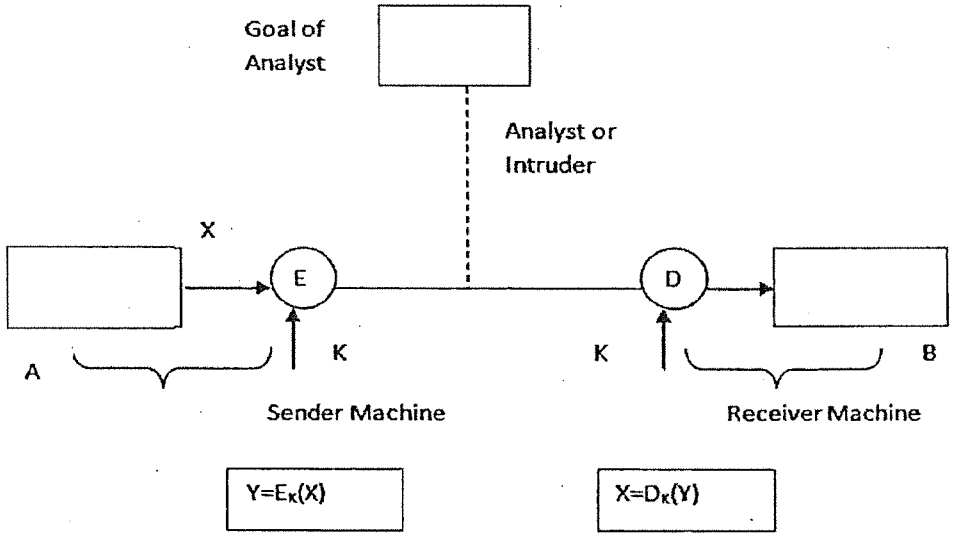
1. Symmetric key: In symmetric key algorithms, the encryption and decryption keys are known both to sender and receiver. The encryption key is shared and the decryption key is easily calculated from it. In many cases, the encryption and decryption keys are the same.

2. Public key: In public key cryptography, encryption key is made public, but it is computationally infeasible to find the decryption key without the information known to the receiver.

A Model for Network Security:



Model for Network Security



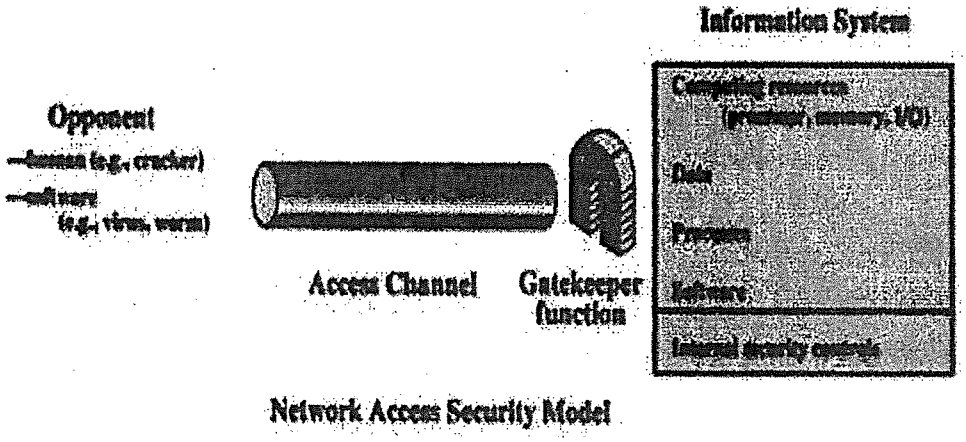
The requirement in this scheme both sender and receiver should know the key in the same way.

Explanation:

1. **Plain Text**: This is the general English language which can be understood by any person.
2. **Cipher Text**: This is resultant after applying encryption algorithm on the plain text with respect to the key.

3. **Encryption**: This is the process by which the plain text is converted into the cipher text using the key.
4. **Decryption**: This is the process by which the cipher text is converted into the plain text.
5. **Key:** This is the secret code used by authorized person while encryption and decryption procedure is doing.
6. **Process in Sender Machine**: In the sender machine the plain text (X) is converted into the cipher text(Y) using a key (K) and it is being processed into the communication channel. This can be written as $Y=E_K(X)$.
7. **Process in Receiver Machine**: In the receiver machine the cipher text(Y) is decrypted into the plain text(X) by using the key. Thus the plain text will be received by the receiver. This can be written as $X=D_K(Y)$.

Network Access Security Model:



Network Access Security Model

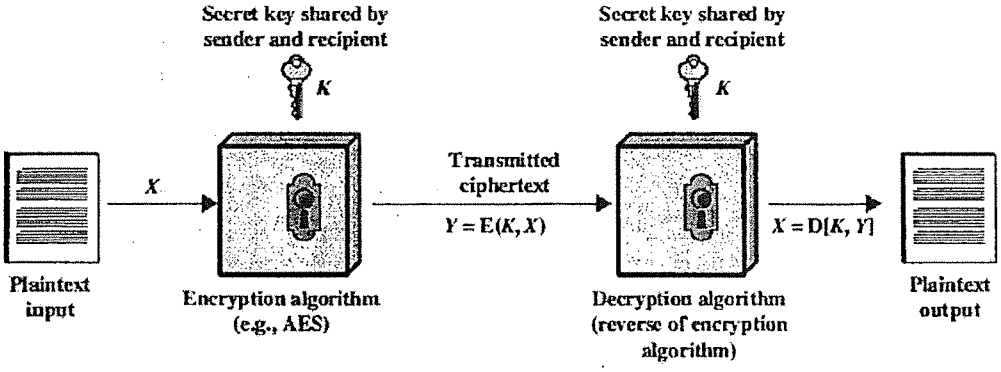
Trusted computer systems can be used to implement this model. By Using this model requires us to:

1. Select appropriate gatekeeper functions to identify users.
2. Implement security controls to ensure only authorized users access designated information or resources.

Conventional Encryption:

This is referred as conventional / private-key / single-key. The Sender and Recipient share a common key. All classical encryption algorithms are private-key was only type prior to invention of public key in 1970. The **plaintext** is known as **original message**. The following basic terminologies used:

1. **Cipher Text** - the coded message
2. **Cipher** - algorithm for transforming plaintext to cipher text
3. **Key** - info used in cipher known only to sender/receiver
4. **Encipher (encrypt)** - converting plaintext to cipher text
5. **Decipher (decrypt)** - recovering cipher text from plaintext
6. **Cryptography** - study of encryption principles/methods
7. **Cryptanalysis (code breaking)** - the study of principles/ methods of deciphering cipher text *without* knowing key
8. **Cryptology** - the field of both cryptography and cryptanalysis



Here the original message, referred to as plaintext, is converted into apparently random nonsense, referred to as cipher text. The encryption process consists of an algorithm and a key.

The key is a value independent of the plaintext. Changing the key changes the output of the algorithm. Once the cipher text is produced, it may be transmitted. Upon reception, the cipher text can be transformed back to the original plaintext by using a decryption algorithm and the same key that was used for encryption. The security depends on several factors. First, the encryption algorithm must be powerful enough that it is impractical to decrypt a message on the basis of cipher text alone. Beyond that, the security depends on the secrecy of the key, not the secrecy of the algorithm.

Two requirements for secure use of symmetric encryption:

a. A strong encryption algorithm

b. A secret key known only to sender / receiver

$Y = E_K(X)$ $X = D_K(Y)$

- assume encryption algorithm is known
- implies a secure channel to distribute key

A source produces a message in plaintext, $X = [X_1, X_2 \ldots X_M]$, where M are the number of letters in the message. A key of the form $K = [K_1, K_2 \ldots K_J]$ is generated. If the key is generated at the source, then it must be provided to the destination by means of some secure channel.

With the message X and the encryption key K as input, the encryption algorithm forms the cipher text $Y = [Y_1, Y_2, Y_N]$. This can be expressed as

$$Y = E_K(X)$$

The intended receiver, in possession of the k e y , is able to invert the transformation:

$$X = D_K(Y)$$

An opponent, observing Y but not having access to K or X, may attempt to recover X or K or both. It is assumed that the opponent knows the encryption and decryption algorithms.

If the opponent is interested in only this particular message, then the focus of effort is to recover X by generating a plaintext estimate. Often if the opponent is interested in being able to read future messages as well, in which case an attempt is made to recover K by generating an estimate.

Ethical and Legal Issues in Computer Security System:

Most ethical and legal issues in computer system are in the area of individual's right to privacy versus the greater good of a larger entity i.e. a company or a society. For example, tracking how employees use computers, crowd surveillance, managing customer profiles, tracking a person's travel with passport and so on. A key concept in resolving this issue is to find out, what is a person's expectation of privacy. Classically, the ethical issues in security system are classified into following 4 categories:

1. **Privacy:** This deals with the right of an individual to control personal information. It is the protection of personal or sensitive information. Privacy is subjective. Different people have different ideas of what privacy is and how much privacy they will trade for safety or convenience.
2. **Accuracy:** This talks about the responsibility for the authenticity, loyalty an accuracy of the information.
3. **Property:** This determines who the owner of the information is and who controls access.

4. **Accessibility:** This deals with the issue of the type of information, an organization has the right to collect. And in that situation, it also expects to know the measures which will safeguard against any unforeseen eventualities.

When dealing with legal issues, we need to remember that there is hierarchy of regulatory bodies that govern the legality of information security. The hierarchy can be roughly described as follows:

International: e.g. International Cybercrime Treaty

Federal: e.g. FERPA, GLB, HIPAA

State: e.g. UCITA, SB 1386 etc.

Organization: e.g. Computer Use policy

Phishing:

Phishing is a form of fraud in which an attacker masquerades as a reputable entity or person in email or other communication channels. The attacker uses phishing emails to distribute malicious links or attachments that can perform a variety of functions, including the extraction of login credentials or account information from victims.

Phishing is popular with cybercriminals, as it is far easier to trick someone into clicking a malicious link in a seemingly legitimate phishing email than trying to break through a computer's defenses.

Ex: The attacker decides to create his/her own Web site, which looks very identical to a real Web site. For example, the attacker can clone Citibank's Web site. The cloning is so clever that the human eye will not be able to distinguish between the real (Citibank's) and fake (attacker's) site.

The attack can use many techniques to attack the bank's customer. The attacker sends an email to the legitimate customers of the bank. The email itself appears to have come from the bank. For ensuring this, the attacker exploits.

The email system to suggest that the sender of the email is some bank official (e.g. accountmanager@citibank.com). This fake email warns the user that there has been some sort of attack on Citibank's computer systems and that the bank wants to issue new passwords to all it customers, or verify their existing PINs, etc. For this purpose, the customer is asked to visit a URL mentioned in the same email.

When the customer (i.e. the victim) innocently clicks on the URL specified in the email, he/she is taken to the attacker's site, and not the bank's original site. There, the customer is prompted to enter confidential information, such as his/her password or PIN.

Since the attacker's fake site looks exactly like the original bank site, the customer provides this information. The attacker gladly accepts this information and displays a Thank you to the unsuspecting victim. In the meanwhile, the attacker now uses the victim's password or PIN to access the bank's real site and can perform any transaction as he/she is the victim.

Objectives of Data Security:

In order to maintain data security, we have to protect the data by using certain mechanisms.

Objectives are:

1. To protect the data from software attacks such as viruses and worms.
2. To provide the authentication data from the misusage of data.
3. To send the data from the sender to recipient securely.

Classical Encryption Techniques:

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

Substitution Techniques:

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

Caesar Cipher Algorithm:

This is the simplest substitution technique developed by Caesar. It involves replacing each letter with an alphabet standing 3 positions, further to it. Note that the alphabets are wrapped around (A after Z). a=0 b=1 ... z=25

Here the alphabets are assigned numerical values (a=1, b=2, ..., z=26). So, we can use the following substitutions.

PT	a b c d ..... w x y z
CT	d e f g ..... z a b c

Where PT refers plain text and CT refers cipher text. We can use the following expression for Caesar Cipher algorithm.

$$C = (P + 3) \bmod 26$$

The Caesar Cipher decryption algorithm uses the following expression.

$$P = (C - 3) \bmod 26$$

A slight variation to the Caesar Cipher algorithm is called "Caption Midnight Code". The difference is in Caesar Cipher algorithm the key is always '3', where as in **Captain Midnight Code** the key is any value between **1 and 25**.

The expression for Captain Midnight Code is:

$C = (P + K) \bmod 26$, where k=1 to 25

$P = (C - K) \bmod 26$, where k=1 to 25 (decryption)

In Captain Midnight Code the key space is increased. Here key space is from 1 to 26.

Brute force Attack:

Consider the following plain text.

"Meet me after the party"

The following is the cipher text when k=4 according to Captain Midnight Code.

"qiix qi ejxiv xli tevxc"

The analyst can break this type of cipher text using Brute force attack. Here the analyst try with all possible values of k (k=1 to 25).

This is as shown in below.

K=1 phhw ph diwhu wkh sduwb

K=2 oggv og chvgt vjg rctva

K=3 nffu nf bgufs uif qbsuz

K=4 meet me after the party

K=5 ----------

|

|

|

K=25 qiix qi ejxiv xli tevxc

The attacker can apply Brute force attack when he knows that

a. The encryption and decryption algorithms are Caesar Cipher or Captain Midnight code.
b. The plain text is always understandable.
c. The key space is limited from 1 to 25.

Play fair Algorithm:

This is the simplest multi letter encryption algorithm. It treats diagrams in the plain text as a single unit and converts it into cipher text diagram.

This algorithm is based on a 5 x 5 matrix. Construction by using the key shared between the parties.

Let us use the keyword MONARCHY .

The 5 x 5 matrix is constructed in the following way.

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

Now we use the following rules to encrypt the given message.

i) Repeating plain text letters in the same diagram are separated with a filler letter i.e., balloon into diagrams as shown below.

ba ll oo n

ll is the diagram which contains same plain letters. So, we use a filler letter to separate there.

Now we get ba l$ lo on

ii) Plain text letters in the diagram that fall in the single or same row of the above matrix are replaced with next letters in the same row.(The letters are wrapped around).

Ex: ON becomes NA

FG becomes GI or GJ

LQ becomes PS

HD becomes YC

iii) Plain text letters in the diagram that fall in the same column of the matrix are replaced with next letter in the same column. (The letters are wrapped around).

Ex: HF becomes FP

BI becomes IS or JS

DT becomes KZ

RZ becomes DR

iv) The plain text letters that are not in same row or not in same column or each plain text letter is replaced by the letter that lies in the same row and other letters column.

Ex: MH becomes OC

HK becomes DF

PX becomes SV

By using these rules we can easily get the cipher text corresponding to the given plain text.

Ex: Meet me after the party

clkl cl olkl dzcf osdzb

Strength of playfair cipher:

Playfair cipher is a great advance over simple mono alphabetic ciphers. Since there are 26 letters, 26x26 = 676 diagrams are possible, so identification of individual diagram is more difficult.

Hill Cipher Algorithm:

It is also a Multi letter encryption algorithm. This algorithm takes 'm' plain text letters and substitutes 'm' cipher letters using a key of m x m matrix. The Hill Cipher algorithm uses the following equations to get the cipher text.

For m=3,

$C1=(k_{11}p_1 + k_{12}p_2 + k_{13}p_3) \bmod 26$

$C2=(k_{21}p_1 + k_{22}p_2 + k_{23}p_3) \bmod 26$

$C3=(k_{31}p_1 + k_{32}p_2 + k_{33}p_3) \bmod 26$

$$\begin{pmatrix} C1 \\ C2 \\ C3 \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \bmod 26$$

i.e., C=KP mod 26, In general C=KP

Ex: Let us consider the plain text "Pay more money" and sender uses the key k is

$$k = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix}$$

The Encryption is as given below.

Pay mor emo ney

First we consider the first 3 letters, 'PAY' as a column vector.

$$\begin{pmatrix} P1 \\ P2 \\ P3 \end{pmatrix} = \begin{pmatrix} P \\ A \\ Y \end{pmatrix} = \begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix}$$

Now

$$\begin{pmatrix} C1 \\ C2 \\ C3 \end{pmatrix} = \begin{pmatrix} 17\ 17\ 5 \\ 21\ 18\ 21 \\ 2\ \ 2\ 19 \end{pmatrix} \begin{pmatrix} 15 \\ 0 \\ 24 \end{pmatrix}$$

$$\begin{pmatrix} C1 \\ C2 \\ C3 \end{pmatrix} = \begin{pmatrix} 375 \\ 819 \\ 48 \end{pmatrix} \bmod 26 = \begin{pmatrix} 11 \\ 13 \\ 18 \end{pmatrix} = \begin{pmatrix} L \\ N \\ S \end{pmatrix}$$

Therefore, PAY is substituted with LNS.

Similarly, we can get the cipher text for the remaining plain text.

After receiving the cipher text the cipher text the receiver uses inverse matrix of K for decryption.

The receiver finds K^{-1} in such a way that $K.K^{-1} = K^{-1}.K = I$

The K^{-1}matrix for the above matrix is

$$K^{-1} = \begin{bmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{bmatrix}$$

Once K^{-1} is found receiver uses the following expression in decryption

$$P=D_K{}^{-1}.C \text{ (or) } P=K^{-1}.C$$

Mono-alphabetic Cipher:

A mono-alphabetic substitution cipher, also known as a simple substitution cipher, relies on a fixed replacement structure. That is, the substitution is fixed for each letter of the alphabet. Thus, if "a" is encrypted to "R", then every time we see the letter "a" in the plaintext, we replace it with the letter "R" in the ciphertext.

A simple example is where each letter is encrypted as the next letter in the alphabet: "a simple message" becomes "B TJNQMF NFTTBHF". In general, when performing a simple substitution manually, it is easiest to generate the ciphertext alphabet first, and encrypt by comparing this to the plaintext alphabet. The table below shows how one might choose to, and we will, lay them out for this example.

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

The ciphertext alphabet for the cipher where we replace each letter by the next letter in the alphabet

There are many different mono-alphabetic substitution ciphers, in fact infinitely many, as each letter can be encrypted to any symbol, not just another letter.

Homophonic Substitution Cipher:

The Homophonic Substitution cipher is a substitution cipher in which single plaintext letters can be replaced by any of several different ciphertext letters. They are generally much more difficult to break than standard substitution ciphers.

The number of characters each letter is replaced by is part of the key, e.g. the letter 'E' might be replaced by any of 5 different symbols, while the letter 'Q' may only be substituted by 1 symbol.

The easiest way to break standard substitution ciphers is to look at the letter frequencies, the letter 'E' is usually the most common letter in English, so the most common ciphertext letter will probably be 'E' (or perhaps 'T'). If we allow the letter 'E' to be replaced by any of 3 different characters, then we can no longer just take the most common letter, since the letter count of 'E' is spread over several characters. As we allow more and more possible alternatives for each letter, the resulting cipher can become very secure.

Ex: Let cipher alphabet is as follows:

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
D	X	S	F	Z	E	H	C	V	I	T	P	G	A	Q	L	K	J	R	U	O	W	M	Y	B	N
9			7				3					5	0					4	6						
			2																						
			1																						

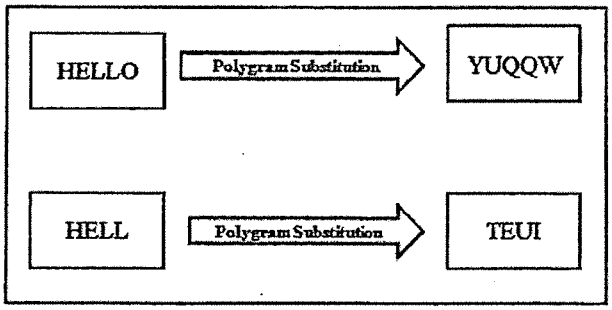
To encipher the message "DEFEND THE EAST WALL OF THE CASTLE", we find 'D' in the top row, and then replace it with the letter below it, 'F'. The second letter, 'E' provides us with several choices, we could use any of 'Z', '7', '2' or '1'. We choose one of these at random, say '7'. After continuing with this, we get the ciphertext:

```
plaintext:  DEFEND THE EAST WALL OF THE CASTLE
ciphertext: F7EZ5F UC2 1DR6 M9PP 0E 6CZ SD4UP1
```

The number of ciphertext letters assigned to each plaintext letter was chosen to flatten the frequency distribution as much as possible. Since 'E' is normally the most common letter, it is allowed more possibilities so that the frequency peak from the letter 'E' will not be present in the ciphertext.

Polygram Substitution Cipher:

This technique replaces one block of plain text with a block of cipher text. It does not work on a character-by-character basis. This is shown in the following figure:



In the above figure, HELLO could be replaced by YUQQW, but HELL could be replaced by a totally different cipher text block TEUI. This shows that in this technique the replacement of plain text happens block-by-block, rather than character-by-character.

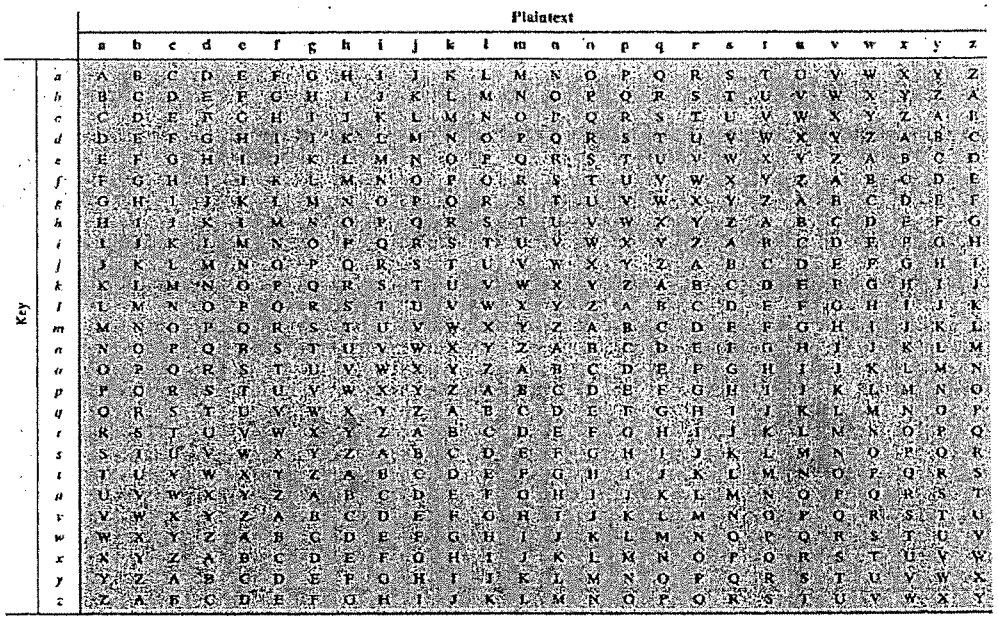
Polyalphabetic Ciphers:

To improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is **polyalphabetic substitution cipher**. All these techniques have the following features in common:

1. A set of related mono-alphabetic substitution rules is used.
2. A key determines which particular rule is chosen for a given transformation.

The best known, and one of the simplest, such algorithm is referred to as the ***Vigenère cipher***. In this scheme, the set of related mono-alphabetic substitution rules consists of the 26 Caesar ciphers, with shifts of 0 through 25. Each cipher is denoted by a key letter, which is the ciphertext letter that substitutes for the plaintext letter a. Thus, a Caesar cipher with a shift of 3 is denoted by the key value d.

The following matrix is known as the ***Vigenère tableau*** which provides relation between key and plain text letter that can be used to produce ciphertext.



Ex: If the keyword is ***deceptive***, the message **"we are discovered save yourself"** is encrypted as follows:

key: *deceptivedeceptivedeceptive*

plaintext: wearediscoveredsaveyourself

ciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

One-Time Pad (Vernam Cipher):

In cryptography, a **One-Time Pad (OTP)** is an encryption technique that cannot be cracked if used correctly. Each new message requires a new key of the same length as the new message is known as a **one-time pad**, is unbreakable. It produces random output that bears no statistical relationship to the plaintext.

Transposition Techniques:

In cryptography, a **transposition cipher** is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the **ciphertext** constitutes a permutation of the plaintext.

Types of Transposition Techniques:

There are three types of transposition techniques. They are

1. Rail Fence Cipher
2. Columnar transposition
3. Double Transposition

1. Rail Fence Cipher:

The plain text is written down as a sequence of diagonals and then read off as a sequence of rows.

Ex: Let the plain text as "meet me after the party".

We can write the plain text with a rail fence as the following:

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

The encrypted message is

MEMATRHTGPRYETEFETEOAAT

2. Columnar Transposition:

The message is written out in rows of a fixed length, and then read out again column by column, and the columns are chosen in some scrambled order. Both the width of the rows and the permutation of the columns are usually defined by a keyword.

For example, the word ZEBRAS is of length 6 (so the columns are of length 6), and the permutation is defined by the alphabetical order of the letters in the keyword. In this case, the order would be "6 3 2 4 1 5".

In a regular columnar transposition cipher, any spare spaces are filled with nulls; in an irregular columnar transposition cipher, the spaces are left blank. Finally, the message is read off in columns, in the order specified by the keyword. For example, suppose we use the keyword ZEBRAS and the message WE ARE DISCOVERED FLEE AT ONCE. In a regular columnar transposition, we write this into the grid as:

```
6 3 2 4 1 5
W E A R E D
I S C O V E
R E D F L E
E A T O N C
E Q K J E U
```

The above matrix consist five nulls (QKJEU) at the end. Now the cipher text is:

EVLNE ACDTK ESEAQ ROFOJ DEECU WIREE

In the irregular case, the columns are not completed by nulls:

```
6 3 2 4 1 5
W E A R E D
I S C O V E
R E D F L E
E A T O N C
E
```

The cipher text is

EVLNA CDTES EAROF ODEEC WIREE

3. Double Transposition:

A single columnar transposition could be attacked by guessing possible column lengths, writing the message out in its columns (but in the wrong order, as the key is not yet known), and then looking for possible anagrams (An **anagram** is a type of word play, the result of rearranging the letters of a word or phrase to produce a new word or phrase, using all the original letters exactly once; for example *Torchwood* can be rearranged into *Doctor Who*). Thus to make it stronger, a double transposition was often used. This is simply a columnar transposition applied twice. The same key can be used for both transpositions, or two different keys can be used.

As an example, we can take the result of the irregular columnar transposition in the previous section, and perform a second encryption with a different keyword, STRIPE, which gives the permutation "564231":

```
5 6 4 2 3 1
E V L N A C
D T E S E A
R O F O D E
E C W I R E
E
```

The cipher text is: `CAEEN SOIAE DRLEF WEDRE EVTOC`

Diffie–Hellman Key Exchange Algorithm:

In a brief, Diffie-Hellman key exchange algorithm is described as follows:

a. A symmetric key approach
b. Developed by Whitfield Diffie and Martin Hellman
c. Used by a number of commercial products

Algorithm:

1. Select a large prime number q such that (q-1) has a large prime factor
2. Select integer a such that 1<=a<=q and "a mod q, a^2 mod q, a^3 mod q, ...$a^{(q-1)}$ mod q" consists of the integers 1 through q-1 in some permutation.
3. User A generates a private random number X_A. Then User A calculates $Y_A=a^{XA}$ mod q
4. User B generates a private random number X_B. Then User B calculates $Y_B=a^{XB}$ mod q
5. User A generates key $K_A = (Y_B)^{XA}$ mod q
6. User B generates key $K_B = (Y_A)^{XB}$ mod q

The purpose of this algorithm is to exchange the key secretly between sender and receiver and this key is used in subsequent encryptions and decryptions.

First we define a primitive root of a prime number 'q' as one whose powers mod q generates all the integers from 1 to q -1 with some permutation i. e., q is a prime number. If 'a' is a primitive root, then a^1 mod q, a^2 mod q, a^3 mod q, ... are all the integers from 1 to q -1 with some permutation.

Ex: q = 7, a = 3

3^1 mod 7, 3^2 mod 7, 3^3 mod 7, 3^4 mod 7, 3^5 mod 7, 3^6 mod 7

3 2 6 4 5 1

∴ 3 is the primitive root of 7

In the Diffie-Hellman algorithm,

1. We have to consider two global (public) elements 'q' (primitive number)
2. and 'a' (primitive root of q).
3. Each user generates his own secret key like A selects X_A, B selects X_B.
4. Each user calculates his public key in the following way :

$$Y_A = a^{XA} \bmod q$$

$$Y_B = a^{XB} \bmod q$$

5. Each user generates a secret key in the following way :

$$K_A = (Y_B)^{XA} \bmod q$$

$$K_B = (Y_A)^{XB} \bmod q$$

K_A and K_B are identical.

6. The final secret key K_A and K_B are identical in the following way :

$$K_A = (Y_B)^{XA} \bmod q$$
$$= (a^{XB} \bmod q)^{XA} \bmod q$$
$$= (a^{XB})^{XA} \bmod q$$
$$= a^{XB \cdot XA} \bmod q$$
$$= a^{XA \cdot XB} \bmod q$$
$$= (a^{XA})^{XB} \bmod q$$
$$= (a^{XA} \bmod q)^{XB} \bmod q$$
$$= (Y_A)^{XB} \bmod q = K_B$$
$$\therefore K_A = K_B$$

Ex: Let the global values q= 97, a = 5, the private keys $X_A = 36$, $X_B = 58$. Find the key which is exchanged between A and B.

Solution: User A Public Key $Y_A = a^{XA} \bmod q$

$$= 5^{36} \bmod 97 = 50$$

$$Y_B = a^{XB} \bmod q$$

$= 5^{58} \bmod 97$

$= (5^{32} * 5^{16} * 5^{8} * 5^{2}) \bmod 97$

$= (35 * 36 * 6 * 25) \bmod 97$

$= 44$

The secret keys are:

$K_A = (Y_B)^{XA} \bmod q$

$= (44)^{36} \bmod 97 = 75$

$K_B = (Y_A)^{XB} \bmod q$

$= (50)^{58} \bmod 97 = 75$

Here K_A and K_B are identical keys. A can encrypt by using 75, B can decrypt by using 75.

Man in the Middle Attack (MIMA) for Diffie-Hellman Key Exchange Algorithm:

Diffie-Hellman key exchange algorithm can fall victim to the MIMA (or to be politically correct, *Woman-in-the-middle attack*), also called as *Bucket brigade attack*. This comes from the way fire fighters of olden days formed a line between fire and water source and passed full buckets towards the fire and the empty buckets back. This is explained in the following steps: (Assign, $n \leftarrow q$, $g \leftarrow a$, $x \leftarrow X_A$, $y \leftarrow X_B$, $K_1 \leftarrow K_A$, $K_2 \leftarrow K_B$)

1. Alice wants to communicate with Bob securely and therefore, she first wants to do a Diffie-Hellman key exchange with him. For this purpose, she sends the values of **n** and **g** to Bob, as usual. Let n=11 and g=7 (A usual these values will form the basis of Alice's A and Bob's B, which will be used to calculate the symmetric key $K_1 = K_2 = K$).

2. Alice does not realize that attacker Tom is listening quietly 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 (i.e. n = 11 and g = 7). This is shown in the following figure:

Alice	Tom	Bob
n = 11, g = 7	n = 11, g = 7	n = 11, g = 7

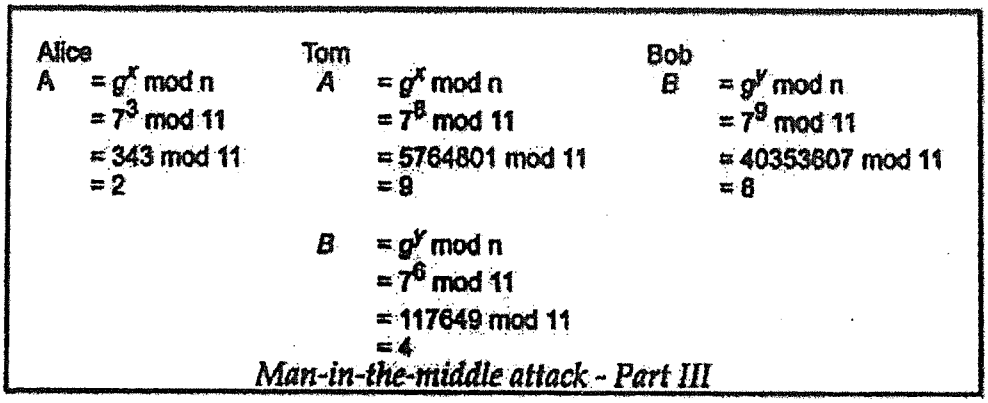
Man in the Middle Attack – Part I

3. Now, let us assume that Alice, Tom and Bob select random numbers **x** and **y** as shown in the following figure:

Alice	Tom	Bob
x = 3	x = 8, y = 6	y = 9

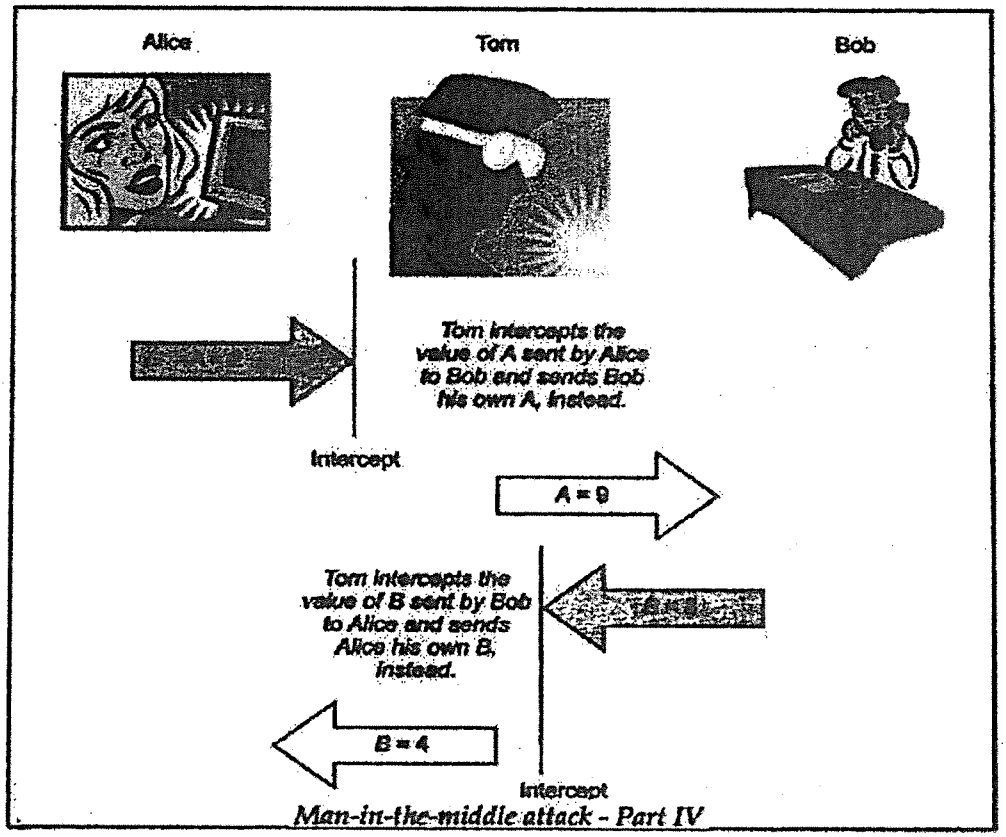
Man in the Middle Attack – Part II

4. Now, based on these values, all three persons calculate the values of A and B as shown in the following figure. Note that Alice and Bob calculate only A and B respectively. However, Tom calculates both A and B.



Man-in-the-middle attack - Part III

5. Now, the real drama begins which is in the following figure:



Man-in-the-middle attack - Part IV

As shown in the above figure the following things happen:

a. Alice sends her (i.e. 2) to Bob. Tom intercepts it and instead given his A (i.e. 9) to Bob. Bob has no idea that Tom has hijacked Alice's A has instead given his A to Bob.

b. In return, Bob sends his B (i.e. 8) to Alice. As before, Tom intercepts it and instead, sends his B (i.e. 4) to Alice. Alice thinks that this B came from Bob. She has no idea that Tom had intercepted the transmission from Bob and changed B.

c. Therefore, at this stage, Alice, Tom Bob has the values of A and B is shown in the following figure:

Alice	Tom	Bob
$A=2, B=4^*$	$A=2, B=8$	$A=9^*, B=8$

(Note: * indicates that these are the values after Tom hijacked and changed them.)

Man-in-the-middle attack - Part V

6. Based on these values, all three persons now calculate their keys as shown in the following figure. We will notice that Alice calculates only K_1, Bob calculates only K_2, whereas Tom calculates both K_1 and K_2.

Alice		Tom		Bob	
K1	$=B^x \bmod n$ $=4^3 \bmod 11$ $=64 \bmod 11$ $=9$	K1	$=B^x \bmod n$ $=8^8 \bmod 11$ $=16777216 \bmod 11$ $=5$	K2	$=A^y \bmod n$ $=9^9 \bmod 11$ $=387420489 \bmod 11$ $=5$
		K2	$=A^y \bmod n$ $=2^6 \bmod 11$ $=64 \bmod 11$ $=9$		

Man-in-the-middle attack - Part VI

As we can see, the MIMA can work against the DH key exchange algorithm, causing it to fail. This is plainly because the man-in-the-middle makes the actual communicators believe that they are talking to each other, whereas they are actually talking to the man-in-the-middle, who is talking to each of them.

This attack can be prevented if Alice and Bob authenticate each other before beginning to exchange information. This provides to Alice is Bob is indeed Bob and not someone else (e.g. Tom) posing as Bob. Similarly, Bob can also get convenience that Alice is genuine as well.

Note: Why Tom needs two Keys? Tom wants to communicate with Alice securely using a shared symmetric key (9) and on the other hand, he wants to communicate with Bob securely using a different shared symmetric key (5). Only then he can receive messages from Alice and Bob, both will(incorrectly) believe that they are directly communicating with each other i.e. Alice will feel that the key 9 is shared between her and Bob, whereas Bob will feel that the key 5 is shared between him and Alice. Actually, what is happening is, Tom sharing the

key 5 with Alice and 5 with Bob. This is also the reason why Tom needed both sets of the secret variables x and y, as well as later on, the non-secret variables A and B.

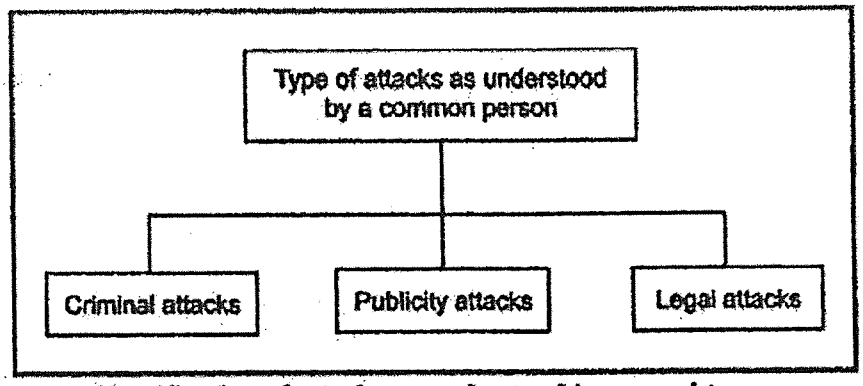
Key Range and Key Size:

The concept of key range and key-size are related to each other. Key Range is total number of keys from smallest to largest available key. An attacker usually is armed with the knowledge of the cryptographic algorithm and the encrypted message, so only the actual key value remains the challenge for the attacker.

1. If the key is found, the attacker can get original plaintext message. In the brute force attack, every possible key in the key-range is tried, until we get the right key.
2. In the best case, the right key is found in the first attempt, in the worst case, the key is found in the last attempt. On an average, the right key is found after trying half of the possible keys in the key-range. Therefore by expanding the key range to a large extent, longer it will take for an attacker to find the key using brute-force attack.
3. The concept of key range leads to the principle of key size. The strength of a cryptographic key is measured with the key size.
4. Key size is measured in bits and is represented using binary number system. Thus if the key range from 0 to 8, then the key size is 3 bits or in other words we can say if the size is bits then the key range is 0 to 256. Key size may be varying, depending upon the applications and the cryptographic algorithm being used, it can be 40 bits, 56 bits, 128 bits & so on. In order to protect the cipher-text against the brute-force attack, the key-size should be such that the attacker cannot crack it within a specified amount of time.
5. From a practical viewpoint, a 40-bit key takes about 3 hours to crack, however a 41-bit key would take 6 hours and 42-bit key would take 12 hours & so on. This means every additional bit doubles the amount of time required to crack the key. We can assume that 128 bit key is quite safe, considering the capabilities of today's computers. However as the computing power and techniques improve, these numbers will change in future.

Types of Attacks:

In a common person's point of view, the attacks are classified into three categories:



Classification of attacks as understood in general terms

1. Criminal Attacks: These attacks are simplest to understand. The main objective of these attackers is to maximize financial gain by attacking computer systems. The different types of criminal attacks are:

a. Fraud: Modern fraud attacks concentrate on manipulating some aspects of electronic currency, credit cards, electronic stock certificates, checks, letters of credit, purchase orders, ATMs, etc.

b. Scams: Scams come in various forms, some of the most common ones being sale of services, auctions, multi-level marketing schemes, general merchandise and business losing their money. A very common example is the Nigeria scam, where an email from Nigeria (and other African countries) entices people to deposit money into a bank account with a promise of hefty gains. Whosoever gets caught in this scam loses money heavily.

c. Destruction: Some sort of grudge is the motive behind such attacks. For example,unhappy employees attack their own organization, Whereas terrorists 'strike at much bigger levels. For example, in the year 2000, there was an attack against popular Internet sites such as yahoo, CNN, eBay, Buy.com, Amazon.com, and e-Trade where authorized users of these sites failed to log in or access these sites.

d. Identity theft: This is best understood with a quote from Bruce Schneider: Why steal from someone when you can just become that person? In other words, an attacker does not steal anything from a legitimate user-he/she becomes that legitimate user': For example, it is much easier to get the password of someone else's bank account, or to actually be able to get a credit card on someone else's name. Then that privilege can be misused until it gets detected.

e. Intellectual property theft: Intellectual property theft ranges from stealing companies' trade secrets, databases, digital music and videos, electronic documents and books, software, and so on.

f. Brand theft: It is quite easy to set up fake Web sites that look like real Web sites, how would a common user known if he/she is visiting the HDFC Bank site or an attacker's site? Innocent users end up providing their secrets and personal details on these fake sites to the attackers. The attackers use these details to then access the real site, causing and identity theft.

2. Publicity Attacks: These attacks occur basically because the attackers wish to see their names on television news channels and newspapers, i.e. gain publicity. These damage the webpages of a site by attacking it. One of the famous such attacks occurred on the on the US Department of Justice's Web site in 1996. The New York Times home page was also famously defaced two years later.

3. Legal Attacks: These attacks are quite unique. Here the attacker tries to make the judge or jury doubtful of the security of a computer system. For example, an attacker may prosecute a bank for a performing an online transaction, which he/she never wanted to perform. In court he/she innocently say something like "The bank's website asked me to enter a password and that is all that I provided; I do not know what happened there after". A judge is likely to sympathize with the attacker.

Possible Types of Attacks:

When the sender of a message encrypts a plain text message into its corresponding cipher text, there are FIVE possibilities for an attack on this message and is explained in the following table:

S. No.	Attack	Things known to the attacker	Things the attacker wants to find out
1	Cipher Text only	a. Cipher text of several messages, all of which are encrypted with the same encryption key b. Algorithm used	a. Plain text messages corresponding to these cipher text messages b. Key used for encryption
2	Known Cipher Text	a. Cipher text of several messages, all of which are encrypted with the same encryption key	a. Algorithm to decrypt cipher text with the same key

		b. Plain text messages corresponding to the above cipher text messages c. Algorithm used	
3	Chosen Plain Text	a. Cipher text and associated plain text messages b. Chooses the plain text to be encrypted	a. Key used for encryption b. Algorithm to decrypt cipher text with the same key
4	Chosen Cipher Text	a. Cipher text of several messages to be decrypted b. Corresponding plain text messages	a. Key used for encryption
5	Chosen Text	a. Some of the above	a. Some of the above

Cookies:

A cookie is a message given to a web browser by a web server. The browser stores the message in a text file. The message is then sent back to the server each time the browser requests a page from the server. Cookies are classified into two types. They are

1. **stateful** – keep track of the previously stored information which is used for current transaction.
2. **stateless** – every transaction is performed as if it were being done for the very first time. There is no previously stored information used for the current transaction.

A **stateless** protocol does not require the server to retain session information or status about each communications partner for the duration of multiple requests. In contrast, a protocol that requires keeping of the internal state on the server is known as a **stateful** protocol.

2. Number Theory

A number of concepts from number theory are essential in the design of public-key cryptographic algorithms.

Prime Numbers:

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

Ex: 2,3,5,7 are prime, 4,6,8,9,10 are not.

Prime Factorisation:

The idea of "factoring" a number is important - finding numbers which divide into it.

a. to **factor** a number n is to write it as a product of other numbers: n=a x b x c
b. factoring a number is relatively hard compared to multiplying the factors together to generate the number
c. the **prime factorisation** of a number n is when its written as a product of primes.
   Ex: $91=7\text{x}13$; $3600=2^4\text{x}3^2\text{x}5^2$; $11011=7\text{x}11^2\text{x}13$
d. If P is the set of all prime numbers, then any positive integer 'a' can be written uniquely in the following form:

$$a = \prod_{p \in P} p^{a_p} \quad \text{where each } a_p \geq 0$$

The right-hand side is the product over all possible prime numbers; for any particular value of, most of the exponents a_p will be 0.

Any integer a > 1 can be factored in a unique way as $a = p_1^{a_1} \times p_2^{a_2} \times \cdots \times p_t^{a_t}$ where $p_1 < p_2 < \cdots < p_t$ are prime numbers and where each is a positive integer. This is known as the fundamental theorem of arithmetic; a proof can be found in any text on number theory.

Relatively Prime Numbers & GCD:

Two numbers a, b are **relatively prime** if there is **no common divisors** apart from 1.

Ex: 8 & 15 are relatively prime since factors of 8 are 1,2,4,8 and of 15 are 1,3,5,15 and 1 is the only common factor

Conversely, we can determine the greatest common divisor (GCD) by comparing their prime factorizations and using least powers.

Ex: $300=2^2\times3^1\times5^2$ $18=2^1\times3^2$ hence $\text{GCD}(18,300)=2^1\times3^1\times5^0=6$

Fermat's Theorem:

If p is prime and a is a positive integer not divisible by p, then

$a^{p-1} \equiv 1 (\text{mod } p) \Rightarrow a^{p-1} \bmod p = 1$, where p is prime and gcd (a, p) = 1.

This also known as Fermat's Little Theorem and is useful in public key cryptography and primality testing.

Ex:

$a = 7, p = 19$
$7^2 = 49 \equiv 11 \text{ (mod19)}$
$7^4 \equiv 121 \equiv 7 \text{ (mod19)}$
$7^8 \equiv 49 \equiv 11 \text{ (mod19)}$
$7^{16} \equiv 121 \equiv 7 \text{ (mod19)}$
$a^{p-1} = 7^{18} = 7^{16} \times 7^2 = 7 \times 11 \equiv 1 \text{ (mod19)}$

Euler's Totient Function:

This function produces 'm' number of positive integer values which are less than 'n' and they are relatively prime to 'n'. This is denoted using the symbol 'ϕ'.

1. $\phi(n) = m$ is the number of positive integers less than n and relatively prime to n.
2. If n is a prime number, then $\phi(n) = n-1$.
3. If n is not a prime, then let p and q are prime numbers, with $p \neq q$, we can write as

$$\phi(n) = \phi(pq) = \phi(p) \times \phi(q) = (p-1) \times (q-1)$$

Ex: a) Determine $\phi(37)$.

n = 37 is a prime number. By using the above function $\phi(37) = 36$. i.e., all the +ve integers from 1 through 36 are relatively prime to 37.

b) Determine $\phi(35)$.

n = 35 is not a prime number. By using the above function, p = 7 and q = 5, 7 * 5 = 35, with $p \neq q$.

$\phi(35) = \phi(7*5) = \phi(7) * \phi(5) = (7-1) * (5-1) = 6 * 4 = 24$ positive integer numbers relatively prime to 35.

Generalisation of Fermat's Theorem:

$a^{\phi(n)} \bmod N = 1$, where gcd (a, N) = 1.

Ex:

i) $a=3; n=10; \phi(10)=4;$
hence $3^4 \bmod 10 = 81 \bmod 10 = 1$

ii) $a=2; n=11; \phi(11)=10;$
hence $2^{10} \bmod 11 = 1024 \bmod 11 = 1$

Euclid's Algorithm – *Finds the greatest common divisor of two integers*

1. The greatest common divisor is the largest integer that evenly divides into both integers.
2. The algorithm repeatedly replaces the original numbers with smaller numbers that have the same greatest common divisor until one of them is zero. The remainder is the greatest common divisor.

This algorithm is based on the following equation: For any non-negative integer ***a*** and any positive integer ***b***,

gcd (a, b) = gcd (b, a mod b)

Ex: gcd (55, 22) = gcd (22, 55 mod 22) = gcd (22, 11) = gcd (11, 22 mod 11) = gcd (11, 0) = 0

Step 1: Select 2 integers **x** and **y** which we believe has a common divisor.

Step 2: Sort x and y in descending order.

Step 3: If y = 0, then return gcd(x, y) = x and stop

Step 4: The remainder is **r = x mod y**

Step 5: Assign x = y and y = r

Step 6: Go to step 3

Ex: gcd(77, 22)

i) x = 77 and y = 22 ≠ 0

ii) r = 77 mod 22 = 11

iii) x = 22 and y = 11 ≠ 0

iv) r = 22 mod 11 = 0

v) x = 11 and y = 0, therefore gcd(77, 22) = x = 11

Extended Euclid's Algorithm:

An extended form of Euclid's algorithm determines the greatest common divisor of two positive integers and, if those numbers are relatively prime, the multiplicative inverse of one with respect to the other (ax + by = gcd (a, b), Where a and b are co-prime, since **x** is the multiplicative inverse of **a** modulo **b**, and **y** is the multiplicative inverse of **b** modulo **a**).

Step 1: (X1, X2, X3) ← (1, 0, f); (Y1, Y2, Y3) ← (0, 1, d)

Step 2: if Y3 = 0 return X3 = gcd (d, f): no inverse

Step 3: if Y3 = 1 return Y3 = gcd (d, f): Y2 = d^{-1} mod f

Step 4: Q = [X3/Y3]

Step 5: (T1, T2, T3) ← (X1 – QY1, X2 – QY2, X3 – QY3)

Step 6: (X1, X2, X3) ← (Y1, Y2, Y3)

Step 7: (Y1, Y2, Y3) ← (T1, T2, T3)

Step 8: go to Step: 2

Ex: The following table is an example of the algorithm. It shows that gcd (550, 1769) = 1 and that the multiplicative inverse of 550 is itself; that is, 550 x 550 = 1 mod 1769.

Extended Euclid (550, 1769)

Q	X1	X2	X3	Y1	Y2	Y3
—	1	0	1769	0	1	550
3	0	1	550	1	−3	119
4	1	−3	119	−4	13	74
1	−4	13	74	5	−16	45
1	5	−16	45	−9	29	29
1	−9	29	29	14	−45	16
1	14	−45	16	−23	74	13
1	−23	74	13	37	−119	3
4	37	−119	3	−171	550	1

Additive Inverse:

The additive inverse is defined as its inverse element under the binary operation of addition, which allows a broad generalization to mathematical objects other than numbers i.e., **additive inverse** of a number a is the number that, when added to a, yields zero. This number is also known as the **opposite** (number), **sign change**, and **negation**. For a real number, it reverses its sign: the opposite to a positive number is negative, and the opposite to a negative number is positive. Zero is the additive inverse of itself.

Ex: The additive inverse of a is denoted by unary minus: $-a$. For example, the additive inverse of 7 is −7, because $7 + (-7) = 0$, and the additive inverse of −0.3 is 0.3, because $-0.3 + 0.3 = 0$.

Multiplicative Inverse:

A **multiplicative inverse** or **reciprocal** for a number x, denoted by $1/x$ or x^{-1}, is a number which when multiplied by x yields the multiplicative identity, 1. The multiplicative inverse of a fraction a/b is b/a. For the multiplicative inverse of a real number, divide 1 by the number. For example, the reciprocal of 5 is one fifth (1/5 or 0.2), and the reciprocal of 0.25 is 1 divided by 0.25, or 4. The **reciprocal function**, the function $f(x)$ that maps x to $1/x$, is one of the simplest examples of a function which is its own inverse.

Modular Multiplicative Inverse:

The modular multiplicative inverse is an integer 'x' such that:

$ax \equiv 1 \pmod{m}$ can be written as $ax \bmod m = 1$, where a and m are integers.

The value of 'x' should be in {0, 1, 2, ... m-1}, i.e., in the ring of integer modulo m.

The multiplicative inverse of "a modulo m" exists if and only if a and m are relatively prime (i.e., if gcd(a, m) = 1).

Ex:

1. Input: a = 3, m = 11

   Output: 4

   Since (4*3) mod 11 = 1, 4 is modulo inverse of 3

   One might think, 15 also as a valid output as "(15*3) mod 11" is also 1, but 15 is not in ring {0, 1, 2, ... 10}, so not valid.

2. Input: a = 10, m = 17

   Output: 12

   Since (10*12) mod 17 = 1, 12 is modulo inverse of 3

****

UNIT – II

1. **Symmetric Key Cryptographic Algorithms:** Algorithm types and modes-overview of symmetric key cryptography-DES-IDEA-Blowfish-AES-Differential and Linear Cryptanalysis.
2. **Asymmetric Key Cryptographic Algorithms:** Overview of asymmetric key cryptography-RSA algorithm-symmetric and asymmetric key cryptography together-digital signatures.

1. Symmetric Key Cryptographic Algorithms

Block Cipher Principles:

Virtually, all symmetric block encryption algorithms in current use are based on a structure referred to as Fiestel block cipher. For that reason, it is important to examine the design principles of the Fiestel cipher. We begin with a **comparison of stream cipher with block cipher.**

A stream cipher is one that encrypts a digital data stream one bit or one byte at a time. E.g., Vigenere Cipher. **A block cipher** is one in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically a block size of 64 or 128 bits is used.

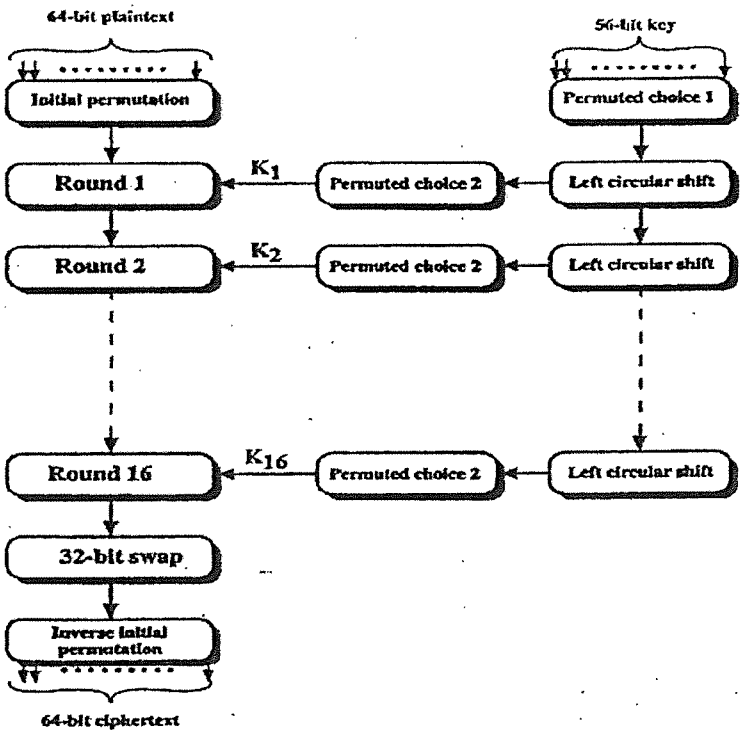
Principles of Block Cipher:

1. Most symmetric block ciphers are based on a **Feistel Cipher Structure** needed since must be able to **decrypt** cipher text to recover messages efficiently. Block ciphers look like an extremely large substitution.
    a. would need table of 264 entries for a 64-bit block
    b. Instead create from smaller building blocks
    c. using idea of a product cipher in 1949 Claude Shannon introduced idea of substitution-permutation (S-P) networks called modern substitution-transposition product cipher these form the basis of modern block ciphers
2. S-P networks are based on the two primitive cryptographic operations we have seen before:
    a. *substitution* (S-box)
    b. *permutation* (P-box)
    c. provide *confusion* and *diffusion* of message

3. **diffusion** – dissipates statistical structure of plaintext over bulk of cipher text
4. **confusion** – makes relationship between cipher text and key as complex as possible

Data Encryption Standard (DES):

The most widely used encryption scheme is based on the DES adopted in 1977 by the National Bureau of Standards (NBS or now NIST-National Institute of Standards and Technology), as Federal Information Standard 46 (FIPS PUB 46). DES algorithm converts 64-bit plain text into 64-bit cipher text using a key of 56 bits. The general diagram for DES Encryption algorithm as follows:



The 64-bit plaintext block is arranged in 8x8 matrix format as the following:

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

The DES encryption algorithm involves five functions. They are

1. Initial Permutation
2. Inverse Initial Permutation

3. 32-bit swap

4. Generation of Sub Keys

5. Details of Single Round

1. Initial Permutation (IP):

The plaintext 64-bit block passed to IP function and the bit positions will be changed. The IP function is described as the following:

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

The output of the IP function is as follows:

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

2. Inverse Initial Permutation (IP^{-1}):

The output of 32-bit swap is passed to IP^{-1} function and the output of IP^{-1} is the 64-bit Cipher text. The format of IP^{-1} function is as follows:

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

3. 32-bit Swap:

The output of 16^{th} round is divided into two 32-bit halves (left half and right half). Now they are swapped and generated 64-bit output.

4. Generation of Sub Keys:

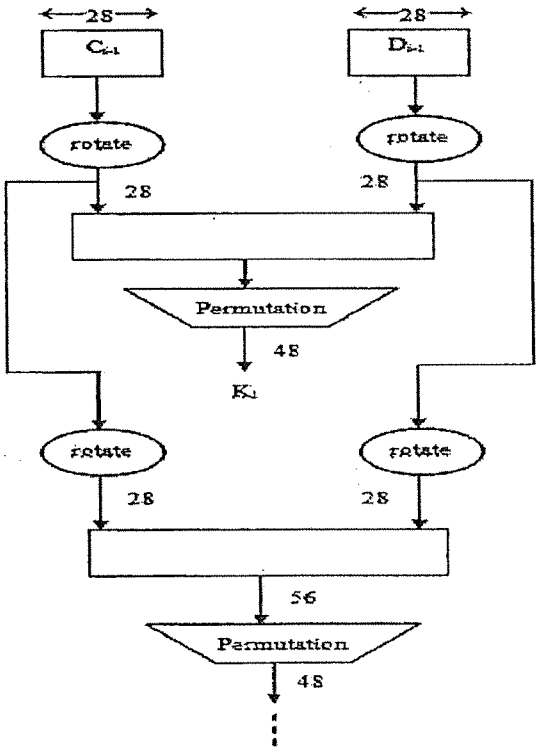
A total of 16 sub keys are required and each one size is 48-bits because there are 16 rounds in DES algorithm. But the input key size is 56-bits.

The actual key of DES algorithm looks like 64-bit long. But each and every 8th bit is called the odd parity. So, we exclude these bits i.e., we exclude 8, 16, 24, 32, 40, 48, 56, 64. From remaining 56-bits generate the 16 sub keys. Initially the 56-bits are permutated and divided into two halves (C_0, D_0) and are explained as the following:

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

C_0 { 57 49 41 33 25 17 9 / 1 58 50 42 34 26 18 / 10 2 59 51 43 35 27 / 19 11 3 60 52 44 36 }

D_0 { 63 55 47 39 31 23 15 / 7 62 54 46 38 30 22 / 14 6 61 53 45 37 29 / 21 13 5 28 20 12 4 }



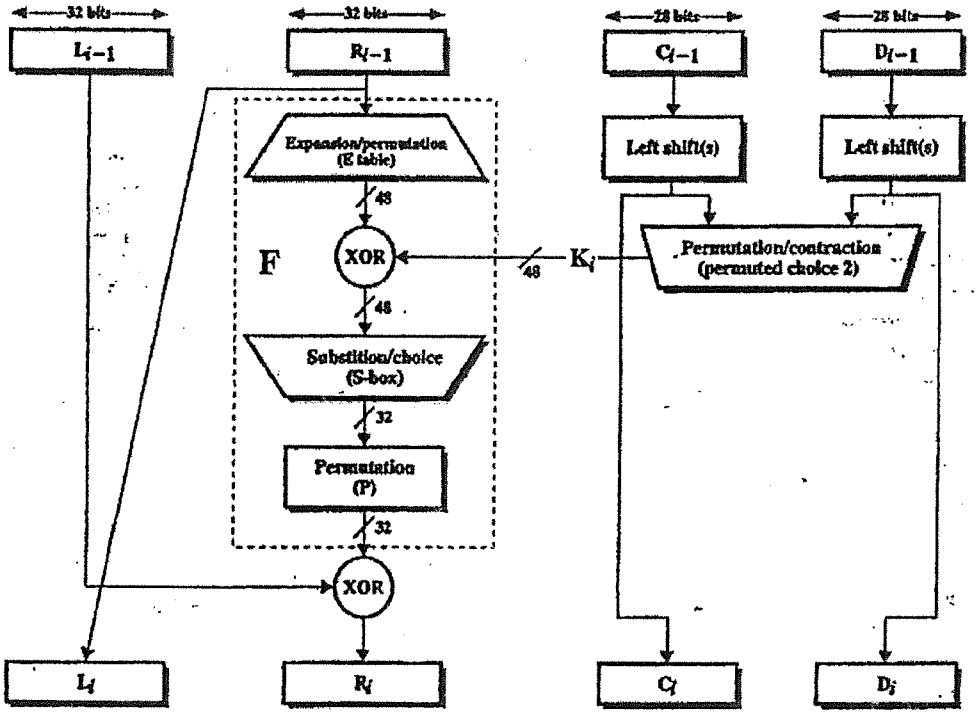
Reduced Permutation:

The two 28-bit blocks are grouped into 56-bit block. This passes through a reduced permutation giving 48-bit block output, representing the key K_i. The general format of reduced permutation function is:

14	17	11	24	1	5	3	28	15	6	21	10
23	19	12	4	26	8	16	7	27	20	13	2
41	52	31	37	47	55	30	40	51	45	33	48
44	49	39	56	34	53	46	42	50	36	29	32

5. Details of Single Round:

The following diagram describes internal structure of a single round. The 64-bit intermediate value is divided into in to 32-bit two halves.



The overall processing at each round can be summarized in the following formulas:

$$L_i = R_{i-1}$$
$$R_i = L_{i-1} \oplus F(R_{i-1}, K_i)$$

The round key K_i is 48 bits. The R input is 32 bits. This R input is first expanded to 48 bits by using a table that defines a permutation plus an expansion that involves duplication of 16 of the R bits. The resulting 48 bits are XORed with K_i.

Expansion Permutation (E)

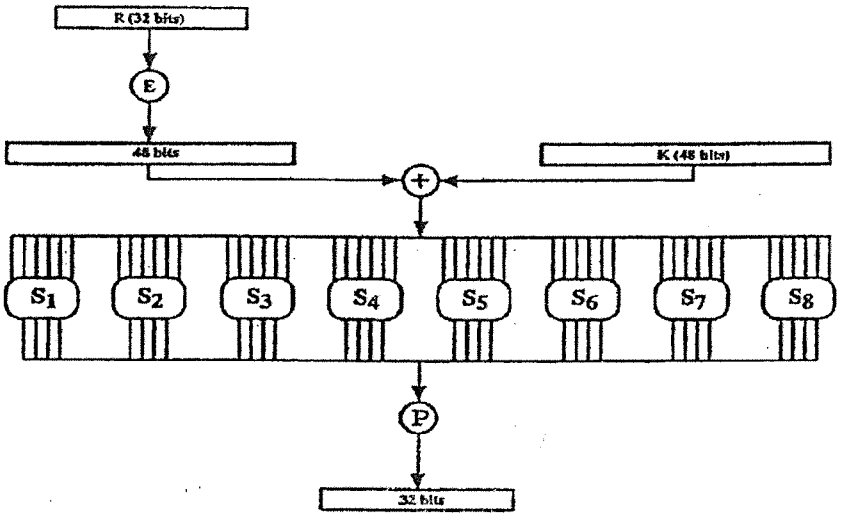
32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

This 48-bit result passes through a substitution function that produces a 32-bit output, which is permuted as defined by the following table.

Permutation Function (P)

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

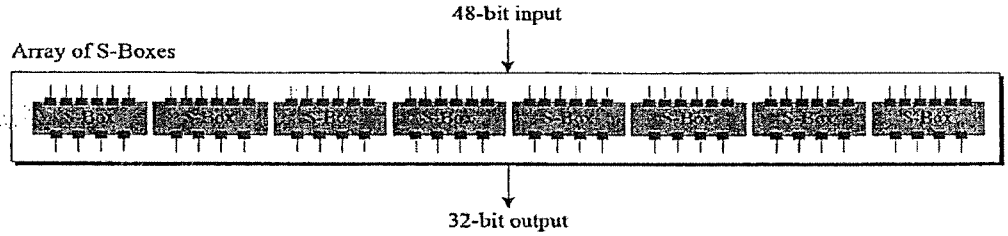
The role of the **S-boxes** in the function **F** is explained in the following diagram. The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output.



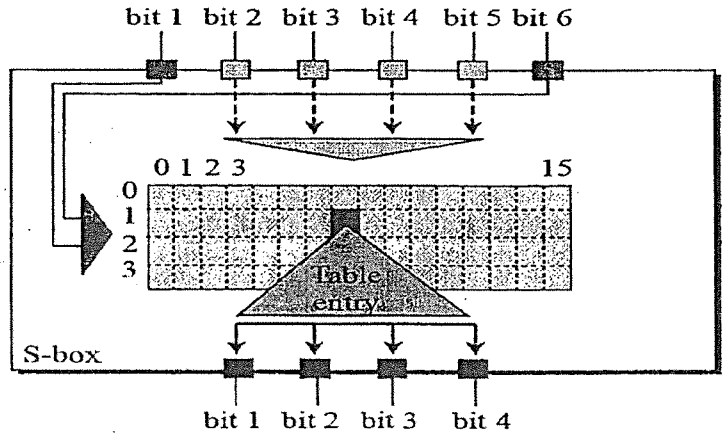
Calculation of F(R, K)

S-Box (Substitution Box):

The S-boxes do the real mixing (confusion). DES uses 8 S-boxes, each with a 6-bit input and a 4-bit output.



S-Box Rule:



The following table shows the permutation for *S-box 1*.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	04	13	01	02	15	11	08	03	10	06	12	05	09	00	07
1	00	15	07	04	14	02	13	10	03	06	12	11	09	05	03	08
2	04	01	14	08	13	06	02	11	15	12	09	07	03	10	05	00
3	15	12	08	02	04	09	01	07	05	11	03	14	10	00	06	13

Ex: The input to S-box 1 is 100011. What is the output?

Solution:

If we write the first and the sixth bits together, we get 11 in binary, which is 3 in decimal. The remaining bits are 0001 in binary, which is 1 in decimal. We look for the value in row 3, column 1, in Table 6.3 (S-box 1). The result is 12 in decimal, which in binary is 1100. So the input 100011 yields the output 1100.

DES Decryption:

The decryption uses the same algorithm as encryption, except that the application of the sub keys is reversed.

The Avalanche Effect:

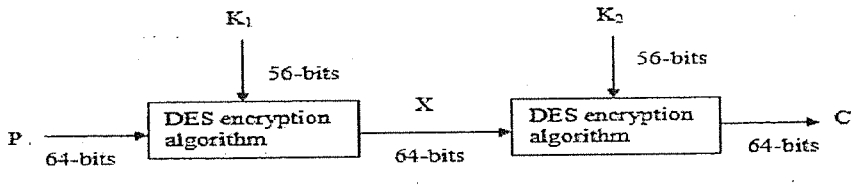
This is a desirable property of any encryption algorithm is that a small change in either the plaintext or the key should produce a significant change in the cipher text. In particular, a change in one bit of the plaintext or one bit of the key should produce a change in many bits of the cipher text.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

The Strength of DES:

With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2×10^{16}. Thus, on the face of it, a brute-force attack appears impractical. Assuming that, on average, half the key space has to be searched, a single machine performing one DES encryption per microsecond would take more than a thousand years to break the cipher.

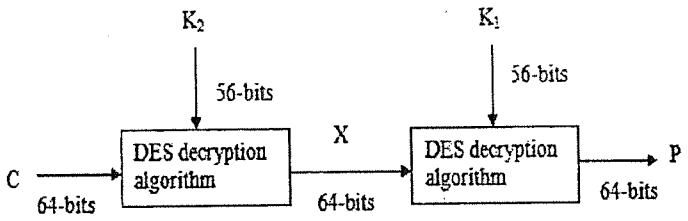
Double DES:

In double DES we have multiple stages of encryption for the given plain text. In the double DES we use encryption twice. The following diagram explains the idea.



$C = E_{K2}(E_{K1}(P))$

Decryption:



$P = D_{K1}(D_{K2}(C))$

This scheme involves a key length of 56 * 2 = 112 bits. It will increase the key space and security.

Suppose it is true that, given two key values it is possible to find a key value K_3 such that

$$E_{K2}(E_{K1}(P)) = E_{K3}(P).$$

If the above equation holds double DES becomes useless and it is equivalent to a Single DES. So, we assume that if DES is used twice as in the above diagram with two different keys, it will produce a cipher text and we will not get that cipher text with any possible single key application of DES i.e., the above equation will not be hold.

Meet in the Middle Attack (MIMA):

Suppose sender and receiver are going to use Double DES algorithm, then the intruder proceeds according to a scheme that does not depend upon any property of DES, but will work. The attack is described as the following.

The intruder observes that $C = E_{K2}(E_{K1}(P))$, $X = E_{K1}(P) = D_{K2}(C)$.

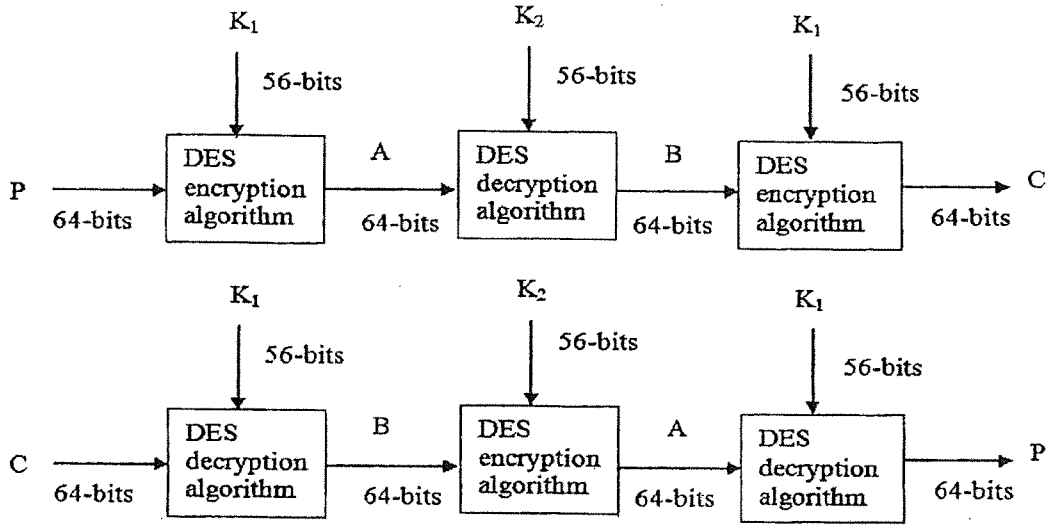
Suppose the intruder knows a pair (P, C). He proceeds in the following way.

The intruder first encrypts the plain text P with all possible 2^{56} keys of K_1. Now he stores all the 2^{56} resultant values in the table and sort them according to their values. Now the intruder decrypts the cipher text with all possible 2^{56} key values of K_2. At the end of each decryption we get a 64 – bit value for X. This value is compared against the table for a match. If the match occurs then the intruder takes the corresponding K_1 and K_2. Now these two keys

are tested against the known (P, C) pair. After he satisfies he will decrypt the remaining cipher text easily.

Triple DES:

The following diagram explains the encryption and decryption used in TDES.



The equations are

$$C = E_{K1}((D_{K2} (E_{K1} (P)))$$

$$P = D_{K1}((E_{K2} (D_{K1} (C)))$$

Block Cipher:

A **block cipher** is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. In general, a block size of 64 or 128 bits is used.

Block Cipher Design Principles:

The block ciphers that is cryptographically strong. There are three critical aspects of block cipher design. They are

a. The number of rounds
b. Design of the function F
c. Key scheduling.

DES Design Criteria:

The criteria used in the design of DES, focused on the design of the S-boxes and on the P function that takes the output of the S boxes. The criteria for the S-boxes are as follows:

1. No output bit of any S-box should be too close a linear function of the input bits. Specifically, if we select any output bit and any subset of the six input bits, the fraction of inputs for which this output bit equals the XOR of these input bits should not be close to 0 or 1, but rather should be near 1/2.

2. Each row of an S-box (determined by a fixed value of the leftmost and rightmost input bits) should include all 16 possible output bit combinations.
3. If two inputs to an S-box differ in exactly one bit, the outputs must differ in at least two bits.
4. If two inputs to an S-box differ in the two middle bits exactly, the outputs must differ in at least two bits.
5. If two inputs to an S-box differ in their first two bits and are identical in their last two bits, the two outputs must not be the same.
6. For any nonzero 6-bit difference between inputs, no more than 8 of the 32 pairs of inputs exhibiting that difference may result in the same output difference.
7. This is a criterion similar to the previous one, but for the case of three S-boxes.

The criteria for the permutation P are as follows:

1. The four output bits from each S-box at round *i* are distributed so that two of them affect (provide input for) "middle bits" of round (*i*+1) and the other two affect end bits. The two middle bits of input to an S-box are not shared with adjacent S-boxes.
2. The end bits are the two left-hand bits and the two right-hand bits, which are shared with adjacent S-boxes.
3. The four output bits from each S-box affect six different S-boxes on the next round, and no two affect the same S-box.
4. For two S-boxes *j*, *k*, if an output bit from S_j affects a middle bit of S_k on the next round, then an output bit from S_k cannot affect a middle bit of S_j. This implies that for $j = k$, an output bit from S_j must not affect a middle bit of S_j.

These criteria are intended to increase the diffusion of the algorithm.

a. The Number of Rounds:

The 16-round DES, a differential cryptanalysis attack is slightly less efficient than brute force: the differential cryptanalysis attack requires $2^{55.1}$ operations, whereas brute force requires 2^{55}. If DES had 15 or fewer rounds, differential cryptanalysis would require less effort than brute-force key search.

b. Design of the Function F:

The function F of DES uses the S-boxes. An n x m S-box typically consists of 2^n rows of m bits each. The n bits of input select one of the rows of the S-box, and the m bits in that row are the output. For larger S-boxes, such as 8 x 32, the question arises as to the best method of selecting the S-box entries in order to meet the type of criteria.

c. Key Scheduling:

A final area of block cipher design is the key schedule algorithm. Hall suggests [ADAM94] that, at minimum, the key schedule should guarantee key/ciphertext Strict Avalanche Criterion and Bit Independence Criterion.

Modes of Operation or Block Cipher Modes:

While encrypting the message of length larger than 64–bits, we use the following block cipher modes.

i) ECB (Electronic Code Book)

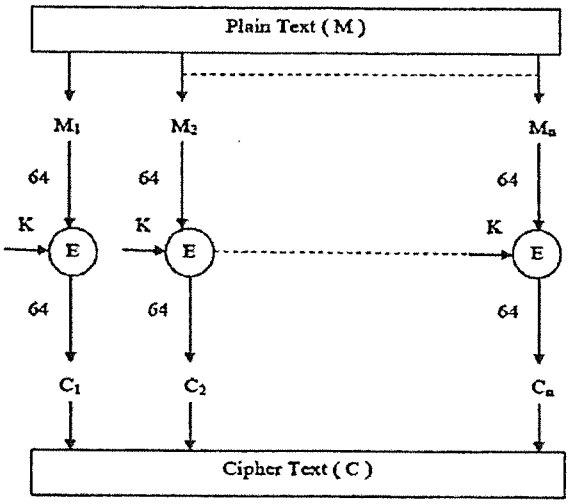
ii) CBC (Cipher Block Chaining)

iii) CFB (Cipher Feedback Mode)

iv) OFB (Output Feedback Mode)

v) Counter Mode

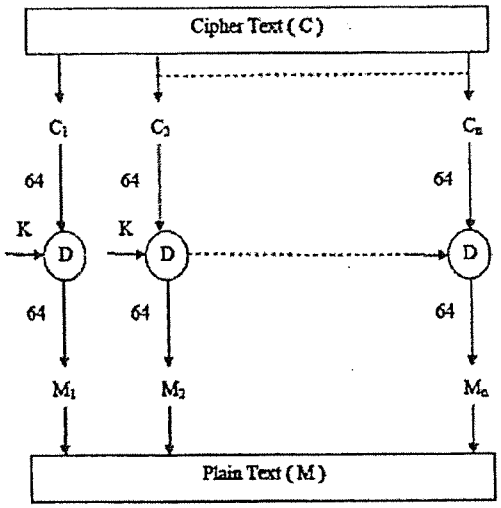
i) ECB: Initially the message is divided into 64–bit blocks. If it is necessary, the last block is padded on right side with 0s (zeros) to get exactly 64-bits. Now each block is encrypted separately with a key, we will get a cipher text block for each plain text block and it is combined to get cipher text of the given message. We have to note that we use the same algorithm and key for each encryption.



In the above diagram,

$$C_1 = E_k [M_1]$$
$$C_2 = E_k [M_2]$$
$$\vdots$$
$$C_n = E_k [M_n]$$

The receiver receives the cipher text. He divides it into 64-bits cipher blocks. Each block is decrypted separately to get plain text blocks. All the blocks are combined to get the original message. The decryption process uses the same key which is used in encryption.

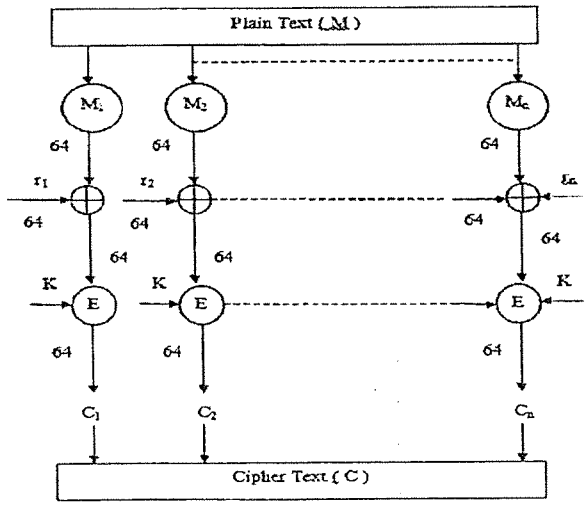


In the above diagram,

$$M_1 = D_k [C_1]$$
$$M_2 = D_k [C_2]$$
$$\vdots$$
$$M_n = D_k [C_n]$$

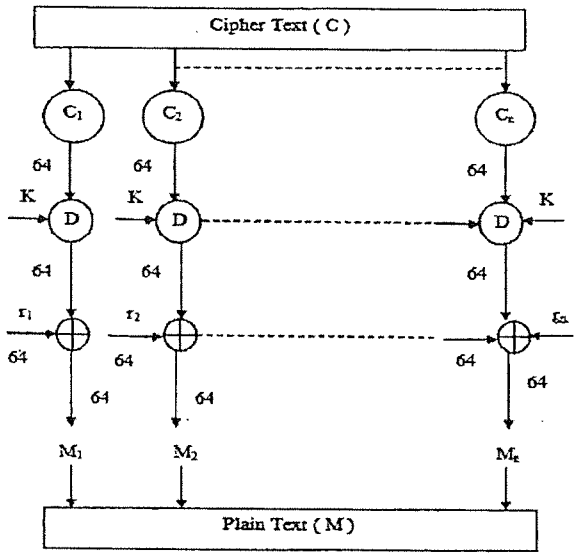
Randomized Electronic Code Book:

In this mode, the drawback in ECB is removed i.e., even if the two plain text blocks are identical, the corresponding cipher blocks are different. This is explained in the following diagram.



$$C_1 = E_k [r_1 \oplus M_1]$$
$$C_2 = E_k [r_2 \oplus M_2]$$
$$\vdots$$
$$C_n = E_k [r_n \oplus M_n]$$

The following is the decryption diagram.



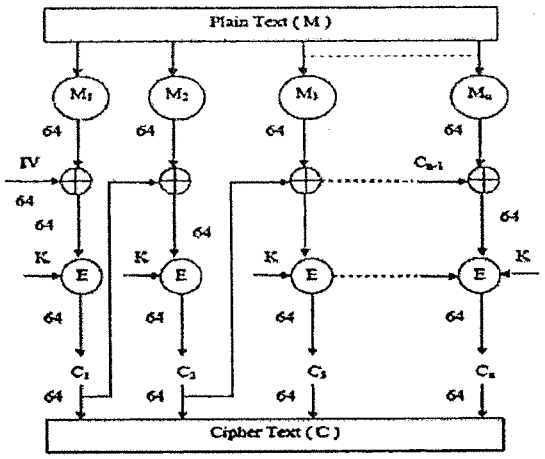
$$M_1 = r_1 \oplus D_k [C1]$$
$$M_2 = r_2 \oplus D_k [C_2]$$
$$\vdots$$
$$M_n = r_n \oplus D_k [C_n]$$

ii) CBC (Cipher Block Chaining) Mode:

CBC is a method of avoiding some of the problem in ECB. In CBC, though the same block repeats in the plain text, it will not cause repeats in the cipher text.

CBC generates its own random numbers. It uses c_i as r_{i+1} i.e., it takes the previous block of cipher text uses it as the random number that will be *Exclusive ored* into the next plain text.

To avoid having two plain text messages that start the same windup with the same cipher text in the beginning, CBC does select one random number, which gets *Exclusive ored* into the first block of plain text and transmits it along with the data. This initial random number is known as an IV (Initialization Vector).



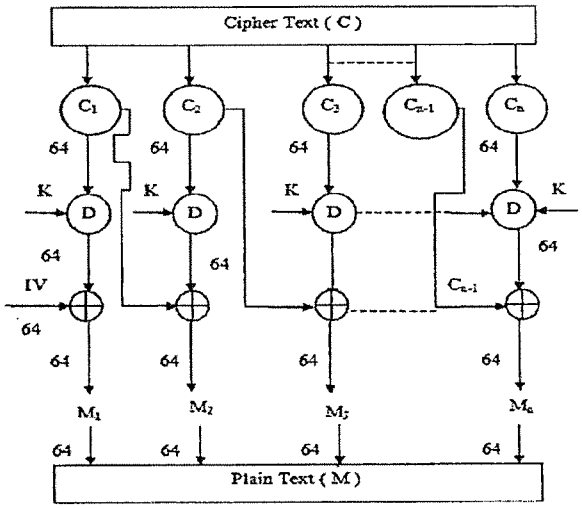
$C_1 = E_k [IV \oplus M_1]$

$C_2 = E_k [C_1 \oplus M_2]$

$C_3 = E_k [C_2 \oplus M_3]$

$C_n = E_k [C_{n-1} \oplus M_1]$

Decryption:



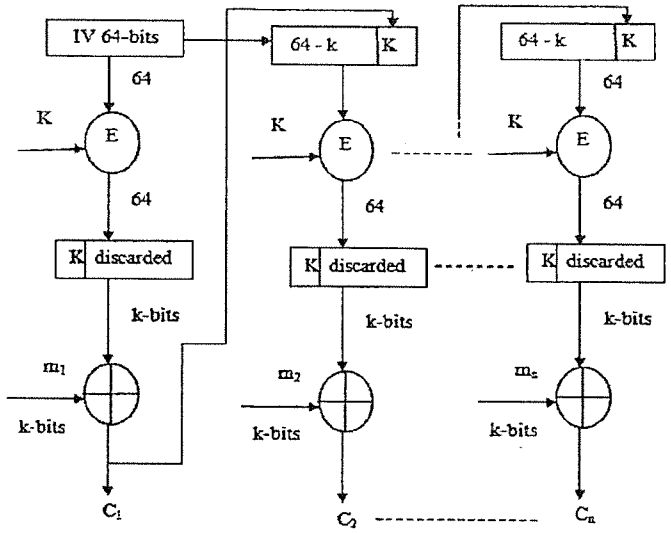
$M_1 = IV \oplus D_k [C_1]$

$M_2 = C_1 \oplus D_k [C_2]$

$M_3 = C_2 \oplus D_k [C_3]$

$M_n = C_{n-1} \oplus D_k [C_n]$

iii) Cipher Feedback Mode (CFB):



$C_1 = M_1 \oplus [S_k [E_k [IV]]]$

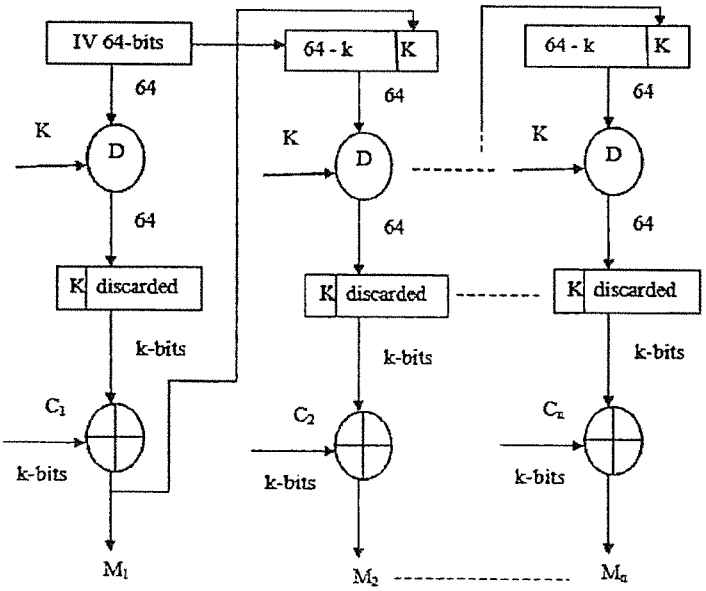
$C_2 = M_2 \oplus [S_k [E_k [IV_{64-k}, C_1]]]$

$C_3 = M_3 \oplus [S_k [E_k [IV_{64-k}, C_2]]]$

$C_n = M_n \oplus [S_k [E_k [IV_{64-k}, C_{n-1}]]]$

In this method, k-bits at a time are generated and Exclusive ored with k-bits of plain text. In CFB, the k-bits shifted in are the k-bits of cipher text from the previous block. So, in CFB the one-time pad cannot be generated before the message is known.

Decryption:



$$M_1 = C_1 \oplus [S_K [D_K [IV]]]$$
$$M_2 = C_2 \oplus [S_K [D_K [IV_{64-k}, M_1]]]$$
$$M_3 = C_3 \oplus [S_K [D_K [IV_{64-k}, M_2]]]$$
$$\vdots$$
$$M_n = C_n \oplus [S_K [D_K [IV_{64-k}, M_{n-1}]]]$$

iv) Output Feedback Mode (OFB):

OFB is a stream cipher. Encryption is performed by the message *Exclusive ored* with the one-time pad generated by OFB.

Let us assume that the stream is created 64-bits at a time. A random number (64-bit) is generated as IV as in CBC mode.

The advantages of this mode is,

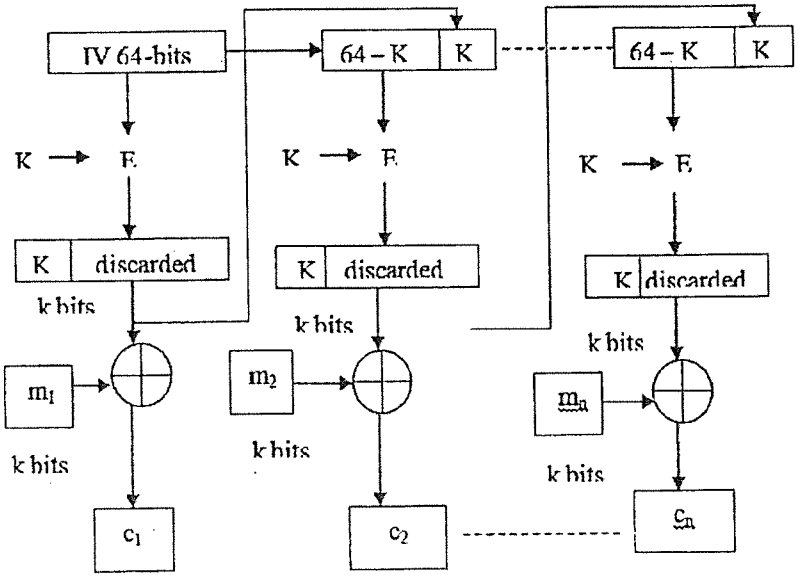
i) The one time pad can be generated in advance, before the message to be encrypted is known.

ii) If some of the bits of the cipher text get garbled, only those bits of plain text get garbled.

iii) A message can arrive in arbitrarily sized and each time a chunk appears, the associated cipher text can be immediately transmitted.

The disadvantage of OFB is,

i) If the plain text and cipher text are known to the bad guy, he can modify the plain text into anything he wants by using *Exclusive ored* with the cipher text with the known plain text and *Exclusive ored* with result with whatever message he wants to transmit.

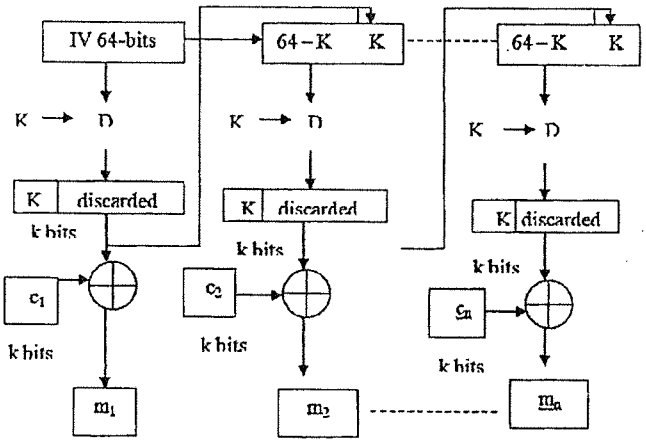


$C_1 = M_1 \oplus [S_k [E_k [IV]]]$

$C_2 = M_2 \oplus [S_k [E_k [IV_{64-k}, S_k]]]$

$C_n = M_n \oplus [S_k [E_k [IV_{64-k}, S_k]]]$

Decryption:



$M_1 = C_1 \oplus [S_k [D_k [IV]]]$

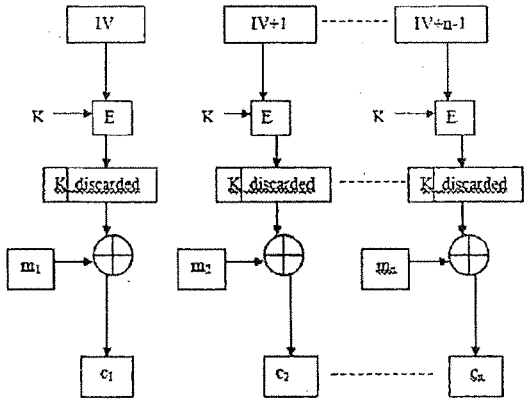
$M_2 = C_2 \oplus [S_k [D_k [IV_{64-k}, S_k]]]$

$M_n = C_n \oplus [S_k [D_k [IV_{64-k}, S_k]]]$

vi) Counter Mode:

Counter mode is similar to Output Feedback Mode in that a onetime pad is generated and *Exclusive ored* with data. It is different in that instead of chaining by encrypting each one-time pad block to get the next one. OFB increments the IV and encrypts the result to get the successive blocks of the one-time pad.

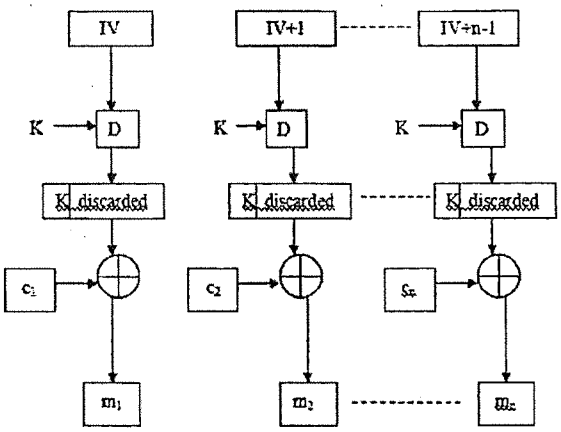
The main advantage of counter mode is that, like OFB, the cryptography can be pre-computed and encryption is simply Excusive or, we can decrypt the message starting at any point rather than being forced to start the beginning.



$$C_1 = M_1 \oplus [S_k [E_k [IV]]]$$
$$C_2 = M_2 \oplus [S_k [E_k [IV + 1]]]$$
$$C_3 = M_3 \oplus [S_k [E_k [IV + 2]]]$$

$$C_n = M_n \oplus [S_k [E_k [IV + n - 1]]]$$

Decryption:



$$M_1 = C_1 \oplus [S_k [D_k [IV]]]$$
$$M_2 = C_2 \oplus [S_k [D_k [IV + 1]]]$$
$$M_3 = C_3 \oplus [S_k [D_k [IV + 2]]]$$

$$M_n = C_n \oplus [S_k [D_k [IV + n - 1]]]$$

Attacks on DES:

The prime concern with DES has been its vulnerability to brute-force attack because of its relatively short (56 bits) key length. However, there has also been interest in 97 finding cryptanalytic attacks on DES. With the increasing popularity of block ciphers with longer key lengths, including triple DES, brute-force attacks have become increasingly impractical. Thus, there has been increased emphasis on cryptanalytic attacks on DES and other symmetric block ciphers are differential cryptanalysis and linear cryptanalysis which are most powerful and promising approaches.

Differential Cryptanalysis:

One of the most significant advances in cryptanalysis in recent years is differential cryptanalysis. Differential cryptanalysis is the first published attack that is capable of breaking DES in less than 2^{55} encryptions. The scheme as reported by Biham and Shamir in 1993, can successfully crypt analyze DES with an effort on the order of 2^{47} encryptions, requiring 2^{47} chosen plaintexts. Although 2^{47} is certainly significantly less than 2^{55}, the need for the adversary to find 2^{47} chosen plaintexts makes this attack of only theoretical interest.

Although differential cryptanalysis is a powerful tool, it does not do very well against DES. The reason is that differential cryptanalysis was known to the team as early as 1974. The need to strengthen DES against attacks using differential cryptanalysis played a large part in the design of the S-boxes and the permutation P. Differential cryptanalysis of an eight-round LUCIFER algorithm requires only 256 chosen plaintexts, whereas an attack on an eight-round version of DES requires 2^{14} chosen plaintexts.

Differential Cryptanalysis Attack:

The differential cryptanalysis attack is complex. Here, we provide a brief overview so that you can get a flavor of the attack. The rationale behind differential cryptanalysis is to observe the behavior of pairs of text blocks evolving along each round of the cipher, instead of observing the evolution of a single text block.

We begin with a change in notation for DES. Consider the original plaintext block **m** to consist of two halves m_0, m_1. Each round of DES maps the right-hand input into the left-hand output and sets the right-hand output to be a function of the left-hand input and the sub-key for this round. So, at each round, only one new 32-bit block is created. If we label each new block m_i, i=2 to 17, then the intermediate message halves are related as follows:

$$m_{i+1} = m_{i-1} + f(m_i, K_i), i = 1, 2, \ldots, 16$$

In differential cryptanalysis, we start with two messages, m and m", with a known XOR difference dm = m ⊕ m', and consider the difference between the intermediate message halves: $dm_i = m_i \oplus m_i'$. Then we have

$$\begin{aligned} dm_{i+1} &= m_{i+1} \oplus m_{i+1}' \\ &= [m_{i-1} \oplus f(m_i, K_i)] \oplus [m_{i-1}' \oplus f(m_i', K_i)] \\ &= dm_{i-1} \oplus [f(m_i, K_i) \oplus f(m_i', K_i)] \end{aligned}$$

Now, suppose that many pairs of inputs to f with the same difference yield the same output difference if the same sub key is used. To put this more precisely, let us say that X may cause Y with probability p, if for a fraction p of the pairs in which the input XOR is X, the output XOR equals Y. We want to suppose that there are a number of values of that have high probability of causing a particular output difference. Therefore, if we know dm_{i-1} and dm_i with high probability, then we know dm_{i+1} with high probability. Furthermore, if a number of such differences are determined, it is feasible to determine the sub-key used in the function f.

The overall strategy of differential cryptanalysis is based on these considerations for a single round. The procedure is to begin with two plaintext messages m and m' with a given difference and trace through a probable pattern of differences after each round to yield a probable difference for the cipher text. Actually, there are two probable patterns of differences for the two 32-bit halves. Next, we submit the plain text for encryption to determine the actual difference under the unknown key and compare the result to the probable difference. If there is a match, then we suspect that all the probable patterns at all the intermediate rounds are correct. With that assumption, we can make some deductions about the key bits. This procedure must be repeated many times to determine all the key bits.

$$E(K, m) \oplus E(K, m') = (dm_{17} || dm_{16})$$

Linear Cryptanalysis:

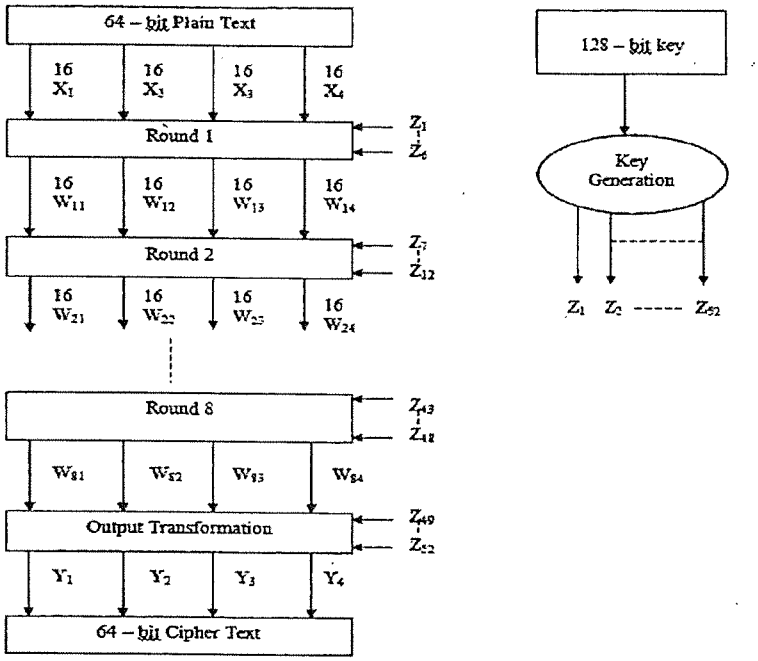
A more recent development is linear cryptanalysis. This attack is based on finding linear approximations to describe the transformations performed in DES. This method can find a DES key given known plaintexts, as compared to chosen plaintexts for differential cryptanalysis. Although this is a minor improvement, because it may be easier to acquire known plaintext rather than chosen plaintext, it still leaves linear cryptanalysis infeasible as an attack on DES. So far, little work has been done by researchers to validate the linear cryptanalytic approach.

International Data Encryption Algorithm (I D E A):

IDEA was originally called **IPES (Improved Proposed Encryption Standard)**. It was developed Xuejia Lai and James L. Massey of ETH Ztiria.

IDEA algorithm converts a 64-bits plain text into 64-bits cipher text using a 128-bits key value. Here the 64-bits plain text undergoes 8 rounds of similar structure and an output transformation function to give 64-bits cipher text.

Here we generate 52 sub keys of 16-bits from the 128-bits key. Each round uses 6 sub keys and the output transformation uses 4 sub keys. The following is the block diagram.



Explanation:

Sub key Generation : First we consider the 128-bits key. It is divided into 8 equal parts. The first part is called Z_1, second part is called Z_2 and so on. The last part is Z_8. In this way we generate Z_1 to Z_8 keys i.e., Z_1 (1 ..16), Z_2 (17 .. 32), Z_3 (33 .. 48), Z_4 (49 .. 64), Z_5 (65 .. 80), Z_6(81 .. 96), Z_7 (97 .. 112), Z_8 (113 .. 128).

Now we perform circular left shift of 25-bits on the given key to get 26 .. 128, 1 .. 25. We divide these bits into 8 equal parts and call them as Z_9 to Z_{16} i.e., Z_9 [26 .. 41], Z_{10} [42 .. 57], Z_{11} [57 .. 73], Z_{12} [74 .. 89], Z_{13} [90 .. 105], Z_{14} [106 .. 121], Z_{15} [122, 123, 124, 125, 126, 127, 128, 1 .. 9], Z_{16} [10 .. 25].

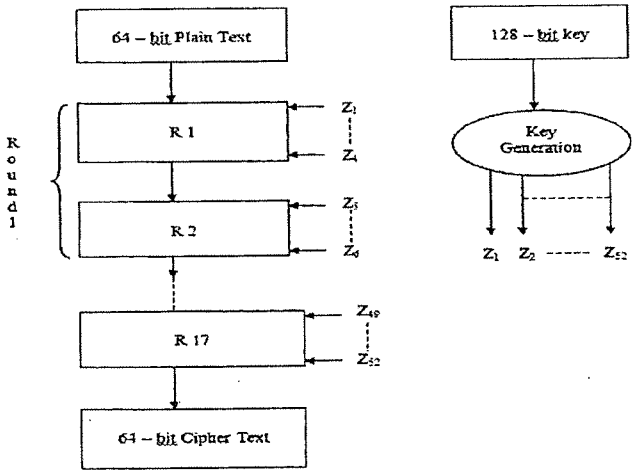
Now again we perform circular left shift of 25-bits on the above input to get

51 .. 128, 1 .. 25, 26 .. 50.

From this we generate the text 8 sub keys from Z_{17} to Z_{24}. Same procedure is repeated until we get 52 sub keys.

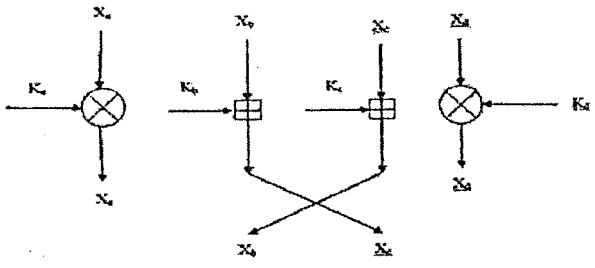
Internal Organization of Rounds:

In IDEA algorithm we have 8 rounds of similar structure and an output transformation function. Each round is further divided into two sub rounds. Hence we will get totally 17 different rounds. Out of these 17 rounds, all rounds are having similar structure. Now the IDEA block diagram can be viewed as follows.



From the above diagram we observe that each round takes a 64-bit input and produces 64-bit input and produces 64-bit output. Each odd round takes 4-keys where as each even round takes 2 keys.

Each round takes four 16-bit values as output. The following is the structure of odd round. The odd round 'i' takes four 16-bit values namely X_a, X_b, X_c and X_d as inputs with keys K_a, K_b, K_c and K_d and produce X_a, X_b, X_c and X_d as outputs. The following diagram explains this concept.

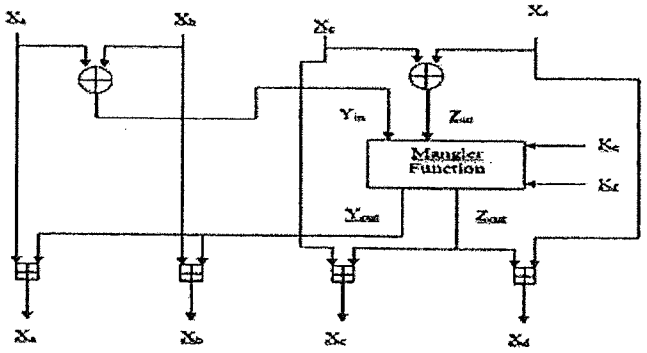


$$X_a = X_a \otimes K_a$$
$$X_b = X_c \boxplus K_c$$
$$X_c = X_b \boxplus K_b$$
$$X_d = X_d \otimes K_d$$

Structure of Even Round:

The even rounds takes four 16-bit inputs X_a, X_b, X_c, X_d and two key values K_e, K_f and generate four 16-bit values output as X_a, X_b, X_c, X_d.

The following diagram explains this concept.



Initially X_a, X_b are *Exclusive ored* to get Y_{in}. X_c, X_d are *Exclusive ored* to get Z_{in}.

i.e., $Y_{in} = X_a \oplus X_b$

$Z_{in} = X_c \oplus X_d$

Now Y_{in}, Z_{in} are given to Mangler function with K_e, K_f. It gives two outputs Y_{out}, Z_{out}.

$Y_{out} = [(K_e \otimes Y_{in}) \boxplus Z_{in}] \otimes K_f$

$Z_{out} = (K_e \otimes Y_{in}) + Y_{out}$

Finally Y_{out} is added to X_a to get new X_a, Y_{out} is added to X_b to get new X_b. Z_{out} is added to X_c to get new X_c, Z_{out} is added to X_d to get X_d.

i.e., $X_a = X_a \boxplus Y_{out}$

$X_b = X_b \boxplus Y_{out}$

$X_c = X_c \boxplus Z_{out}$

$X_d = X_d \boxplus Z_{out}$

IDEA Decryption:

In IDEA decryption the 64-bit cipher text is converted into 64-bits plain text using the same 128-bits key.

The 64-bit cipher text undergoes 8-rounds and output transformation to get 64-bit plain text. In decryption process we require 52 sub keys of 16-bits each. These are labeled as U_1, U_2, U_3, ... U_{52}. These keys are identical to Z_1 to Z_{52} with some permutation; minor changes i.e., the decryption keys are generated by using the encryption keys in the following way.

Stage	Encryption Keys Used	Decryption Keys Used	Equivalent to
R_1	Z_1 Z_2 Z_3 Z_4 Z_5 Z_6	U_1 U_2 U_3 U_4 U_5 U_6	Z_{49}^{-1} $-Z_{50}$ $-Z_{51}$ Z_{52}^{-1} Z_{47} Z_{48}
R_2	Z_7 Z_8 Z_9 Z_{10} Z_{11} Z_{12}	U_7 U_8 U_9 U_{10} U_{11} U_{12}	Z_{43}^{-1} $-Z_{44}$ $-Z_{45}$ Z_{46}^{-1} Z_{41} Z_{42}
R_3	Z_{13} Z_{14} Z_{15} Z_{16} Z_{17} Z_{18}	U_{13} U_{14} U_{15} U_{16} U_{17} U_{18}	Z_{37}^{-1} $-Z_{38}$ $-Z_{39}$ Z_{40}^{-1} Z_{35} Z_{36}
R_4	Z_{19} Z_{20} Z_{21} Z_{22} Z_{23} Z_{24}	U_{19} U_{20} U_{21} U_{22} U_{23} U_{24}	Z_{31}^{-1} $-Z_{32}$ $-Z_{33}$ Z_{34}^{-1} Z_{29} Z_{30}
R_5	Z_{25} Z_{26} Z_{27} Z_{28} Z_{29} Z_{30}	U_{25} U_{26} U_{27} U_{28} U_{29} U_{30}	Z_{25}^{-1} $-Z_{26}$ $-Z_{27}$ Z_{28}^{-1} Z_{23} Z_{24}
R_6	Z_{31} Z_{32} Z_{33} Z_{34} Z_{35} Z_{36}	U_{31} U_{32} U_{33} U_{34} U_{35} U_{36}	Z_{19}^{-1} $-Z_{20}$ $-Z_{21}$ Z_{22}^{-1} Z_{17} Z_{18}
R_7	Z_{37} Z_{38} Z_{39} Z_{40} Z_{41} Z_{42}	U_{37} U_{38} U_{39} U_{40} U_{41} U_{42}	Z_{13}^{-1} $-Z_{14}$ $-Z_{15}$ Z_{16}^{-1} Z_{11} Z_{12}
R_8	Z_{43} Z_{44} Z_{45} Z_{46} Z_{47} Z_{48}	U_{43} U_{44} U_{45} U_{46} U_{47} U_{48}	Z_7^{-1} $-Z_8$ $-Z_9$ Z_{10}^{-1} Z_5 Z_6
Output (R_9)	Z_{49} Z_{50} Z_{51} Z_{52}	U_{49} U_{50} U_{51} U_{52}	Z_1^{-1} $-Z_2$ $-Z_3$ Z_4^{-1}

Note 1:

- $x \oplus x = 0$
- $x \oplus 0 = x$
- $x \oplus 1 = \bar{x}$

Note 2: In IDEA algorithm we are using 3 operations. They are

⊞ *Addition Modulo* – It performs addition of two inputs *modulo* 2^{16}.

⊗ *Multiplicative Modulo* - It performs multiplication of two inputs *modulo* 2^{16}. While multiplying the block of all zeros is considered as 2^{16}.

⊕ *Exclusive OR* – It is simple Exclusive or operation. $x \oplus x = 0$, $x \oplus 0 = x$, $x \oplus 1 = \bar{x}$.

Ex: We will see how these operations can be worked for 2-bits.

X	Y	X ⊕ Y	X ⊞ Y	X ⊗ Y
00	00	00	00	01
00	01	01	01	00
00	10	10	10	10
00	11	11	11	10
01	00	01	01	00
01	01	00	10	01
01	10	11	11	11
01	11	10	00	10
10	00	10	10	11
10	01	11	11	10
10	10	00	00	00
10	11	01	01	01
11	00	11	11	10
11	01	10	00	11
11	10	01	01	01
11	11	00	10	00

Important Definitions:

P-Box:

In **cryptography**, a permutation **box** (or **P-box**) is a method of bit-shuffling used to permute or transpose bits across S-**boxes** inputs, retaining diffusion while transposing.

Diffusion and Confusion:

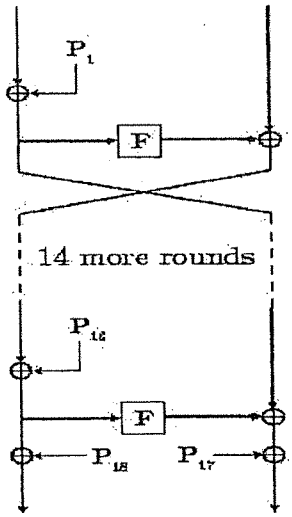
Confusion means that each character of the cipher text should depend on several parts of the key. ***Diffusion*** means that if we change a character of the plaintext, then several characters of the cipher text should change, and similarly, if we change a character of the cipher text, then several characters of the plaintext should change.

Blowfish was designed in 1993 by Bruce Scheier as a fast, alternative to existing encryption algorithms such AES, DES and 3 DES etc.

Blowfish Encryption Algorithm:

Blowfish is a symmetric block encryption algorithm designed in consideration with:

1. **Fast :** It encrypts data on large 32-bit microprocessors at a rate of 26 clock cycles per byte.
2. **Compact:** It can run in less than 5K of memory.
3. **Simple:** It uses addition, XOR, lookup table with 32-bit operands.
4. **Secure:** The key length is variable ,it can be in the range of 32~448 bits: default 128 bits key length.
5. It is suitable for applications where the key does not change often, like communication link or an automatic file encryptor.
6. Unpatented and royality-free.



The Feistel structure of Blowfish

Description of Algorithm:

Blowfish symmetric block cipher algorithm encrypts block data of 64-bits at a time.it will follows the Feistel network and this algorithm is divided into two parts.

a. Key-Expansion

b. Data Encryption

a. Key-Expansion: It will convert a key of at most 448 bits into several sub key arrays totaling 4168 bytes. Blowfish uses large number of sub keys. These keys are generating earlier to any data encryption or decryption.

The p-array consists of 18, 32-bit sub keys:

$P_1, P_2, \ldots\ldots\ldots\ldots, P_{18}$

Four 32-bit S-Boxes consist of 256 entries each:

$S_1,0, S_1,1, \ldots\ldots\ldots S_1,255$

$S_2,0, S_2,1, \ldots\ldots\ldots S_2,255$

$S_3,0, S_3,1, \ldots\ldots\ldots S_3,255$

S4,0, S4,1,.....S4,255

Generating the Sub keys:

The subkeys are calculated using the Blowfish algorithm:

1. Initialize first the P-array and then the four S-boxes, in order, with a fixed string. This string consists of the hexadecimal digits of pi (less the initial 3): P1 = 0x243f6a88, P2 = 0x85a308d3, P3 = 0x13198a2e, P4 = 0x03707344, etc.
2. XOR P1 with the first 32 bits of the key, XOR P2 with the second 32-bits of the key, and so on for all bits of the key (possibly up to P14). Repeatedly cycle through the

key bits until the entire P-array has been XORed with key bits. (For every short key, there is at least one equivalent longer key; for example, if A is a 64-bit key, then AA, AAA, etc., are equivalent keys.)

3. Encrypt the all-zero string with the Blowfish algorithm, using the sub keys described in steps (1) and (2).
4. Replace P1 and P2 with the output of step (3).
5. Encrypt the output of step (3) using the Blowfish algorithm with the modified sub keys.
6. Replace P3 and P4 with the output of step (5).
7. Continue the process, replacing all entries of the P array, and then all four S-boxes in order, with the output of the continuously changing Blowfish algorithm.

In total, 521 iterations are required to generate all required sub keys. Applications can store the sub keys rather than execute this derivation process multiple times.

b. Data Encryption: It is having a function to iterate 16 times of network. Each round consists of key-dependent permutation and a key and data-dependent substitution. All operations are XORs and additions on 32-bit words. The only additional operations are four indexed array data lookup tables for each round.

Algorithm: Blowfish Encryption

Divide x into two 32-bit halves: xL, xR

for i = 1 to 16:

xL = XL XOR Pi

xR = F(XL) XOR xR

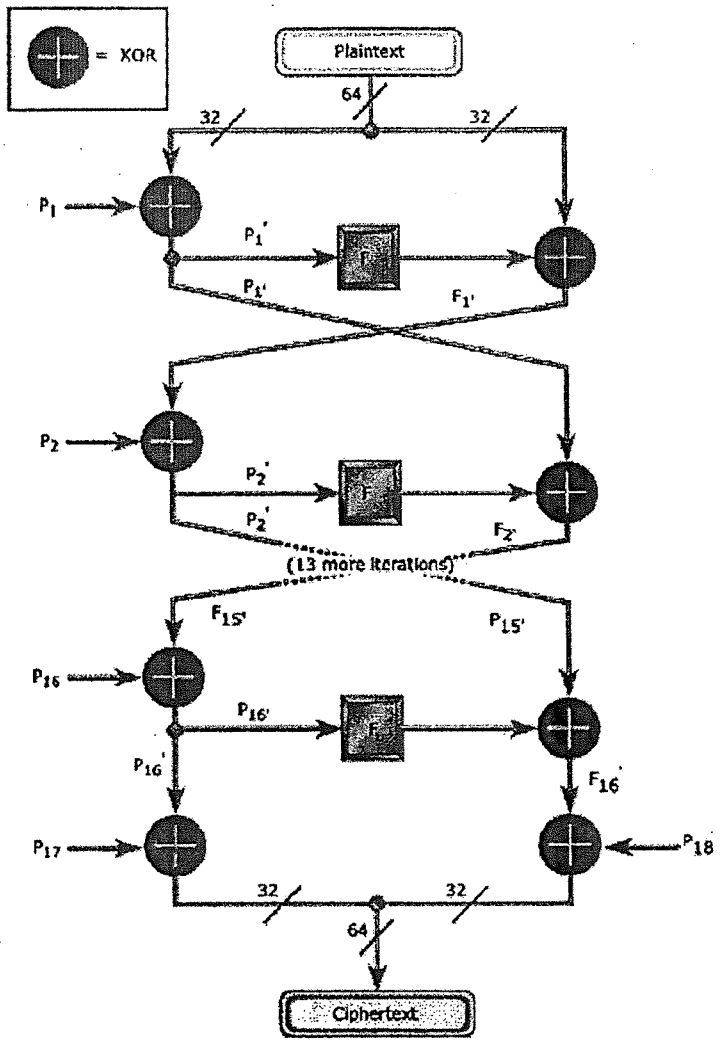
Swap XL and xR

Swap XL and xR (Undo the last swap.)

xR = xR XOR P17

xL = xL XOR P18

Recombine xL and xR



Blowfish Encryption

Advanced Data Encryption Standard (AES):

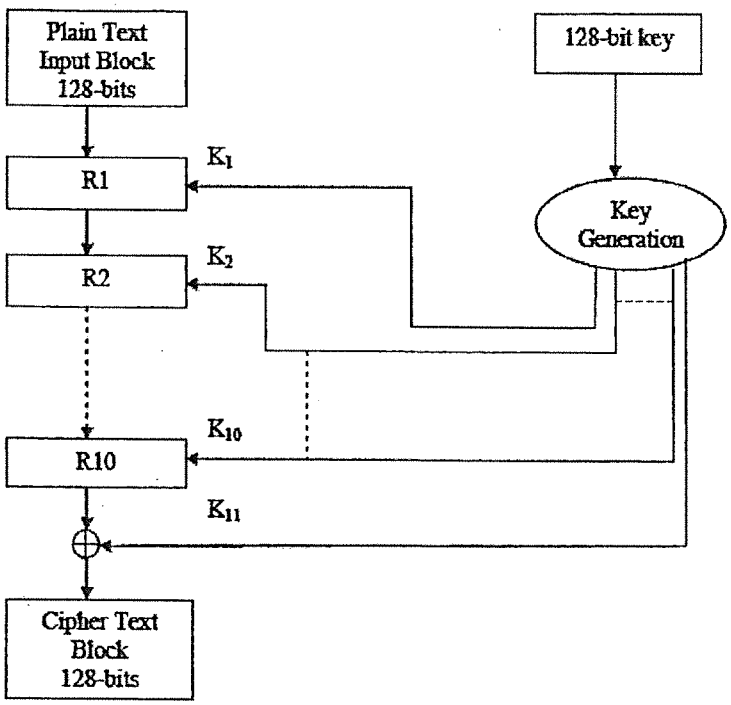
This block cipher algorithm is proposed by Rijndael. This is a substitution permutation network (DES is Fiestal Cipher Network). This is relatively easy to implement when compared to DES and it occupies less memory space. This algorithm uses a block of plain text and converts it into a block of cipher text using a block key.

The block length is a multiple of 32 and in between 128 and 256 bits. This algorithm supports a large key in multiples of 32. The key is also a multiple of 32, between 128 and 256 bits.

The number of rounds in this algorithm depends upon the length of the key. If key length is 128 bits then number of rounds is 10. If key length is 192, we have 12 rounds. If key length is 256, we have 14 rounds.

Block Diagram:

The AES algorithm operates on 4 * 4 arrays of bytes. The following is the block diagram of AES algorithm, which converts 128-bits plain text into 128-bit cipher text using 128-bit key i.e., 10 rounds.



Each round in the AES algorithm contains 4 stages except the last round. They are

i) Sub Bytes Step

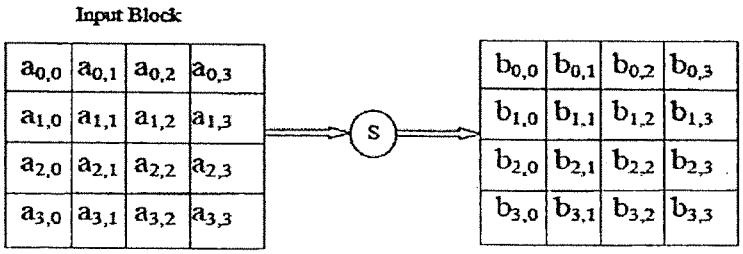
ii) Shift Row Step

iii) Mix Column Step

iv) Add Round key Step

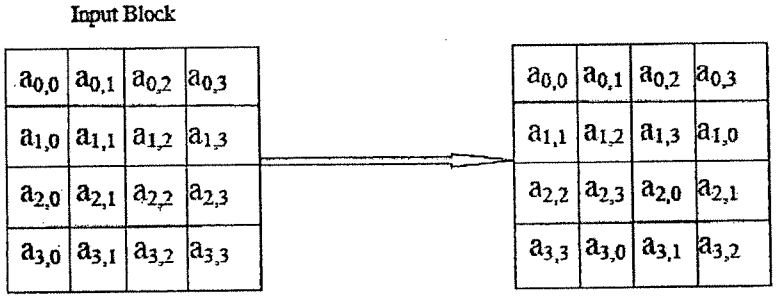
i) Sub Bytes Step:

In sub bytes step each byte of the input is updated using S-box to get an output byte. This operation provides a non-linearity in the input. The S-box is generally derived from the inverse functions. The following diagram explains this idea.



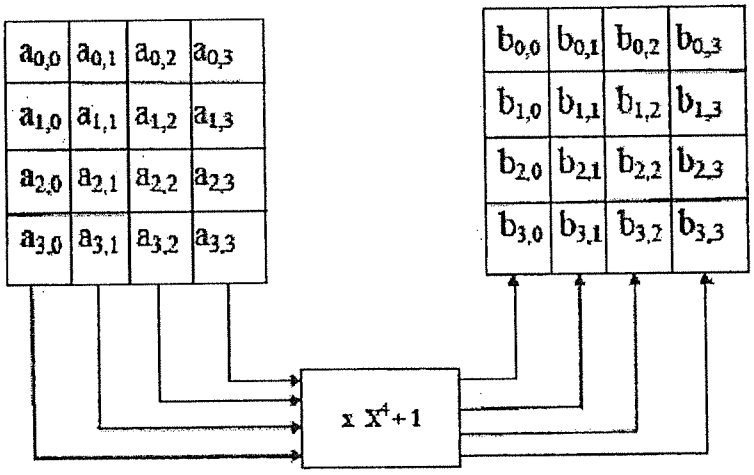
ii) Shift Row Step:

The output of first stage is given to shift rows step. It operates on the rows of the input. The row 1 is unchanged. The second row is shifted one to its left. The third row is shifted two to its left. The fourth row is shifted three to its left. This is shown in the following diagram.



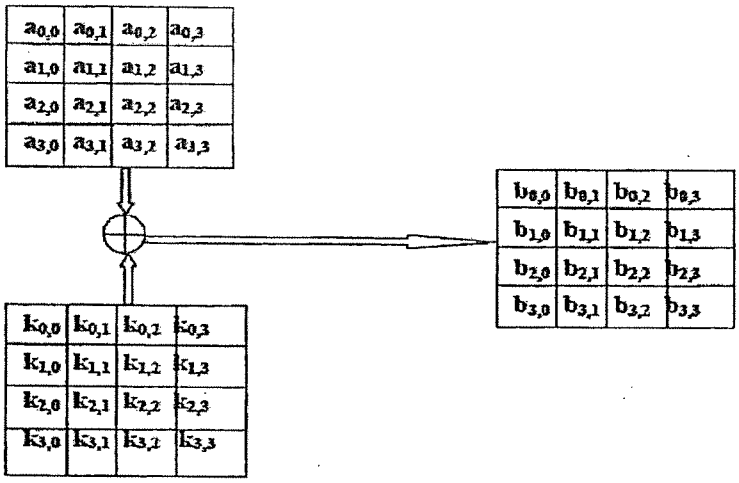
iii) Mix Column Step:

In this step the 4-bytes of each column are converted into transformations and multiply with a fixed polynomial x4 + 1 to get the output. The input to this stage is 4 * 4 array of bytes (output of the shift rows). From this input, we get four groups. Each group is transformed into polynomial. Each such polynomial is *multiplied with* X4 + 1 to get another polynomial. It is again retransformed to a group of 4-bytes.



iv) Add Round Key Step:

In this step sub or round key is combined with each byte of the input to get the output. We require a sub key of 4 * 4 arrays of bytes for each round. These sub keys are generated from the given 128-bits key using key scheduler process. Each byte of the sub key is *Exclusive ored* with corresponding byte of the input to get an output byte. The last round does not have Mix Column Step. But it has the remaining 3 stages in the same order.

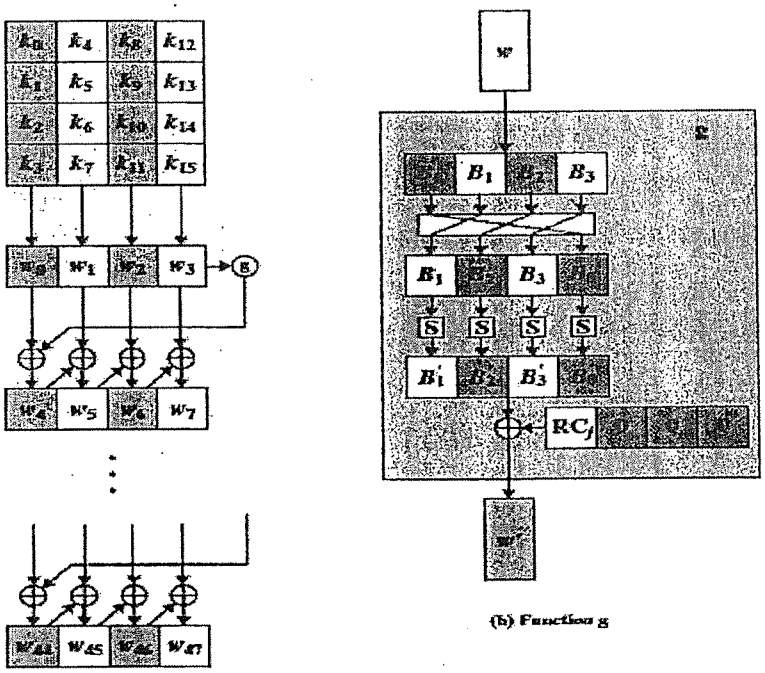


AES Key Expansion or Sub Keys Generation:

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

```
KeyExpansion (byte key[16], word w[44])
{
  word temp;
  for (i = 0; i < 4; i++)
      w[i] = (key[4*i], key[4*i+1], key[4*i+2],    key[4*i+3]);
  for (i = 4; i < 44; i++)
  {
     temp = w[i – 1];
     if (i mod 4 == 0)
        temp = SubWord (RotWord (temp)) ⊕ Rcon[i/4];
     w[i] = w[i–4] ⊕ temp;
  }
}
```

The key is copied into the first four words of the expanded key. The remainder of the expanded key is filled in four words at a time. Each added word w[i] depends on the immediately preceding word, w[i-1], and the word four positions back, w[i-4]. In three out of four cases, a simple XOR is used. For a word whose position in the w array is a multiple of 4, a more complex function is used. The following explains the generation of the expanded key, using the symbol g to represent that complex function. The function g consists of the following sub functions.



(a) Overall algorithm (b) Function g

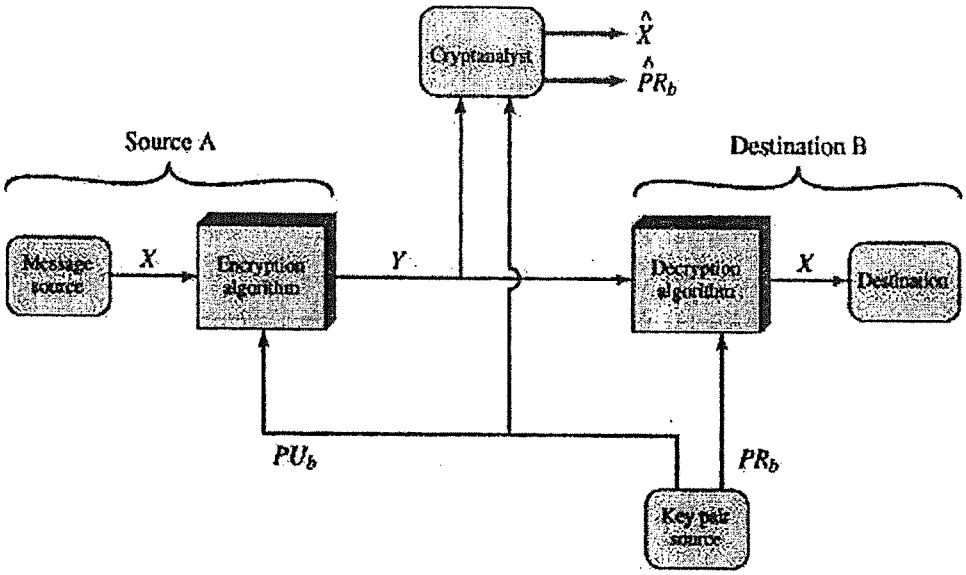
AES Key Expansion

The round constant is a word in which the three rightmost bytes are always 0. Thus, the effect of an XOR of a word with Rcon is to only perform an XOR on the leftmost byte of the word. The round constant is different for each round and is defined as Rcon[j] = (RC[j], 0, 0, 0), with RC[1] = 1, RC[j] = 2 * RC[j–1], and with multiplication defined over the field GF(2^8).

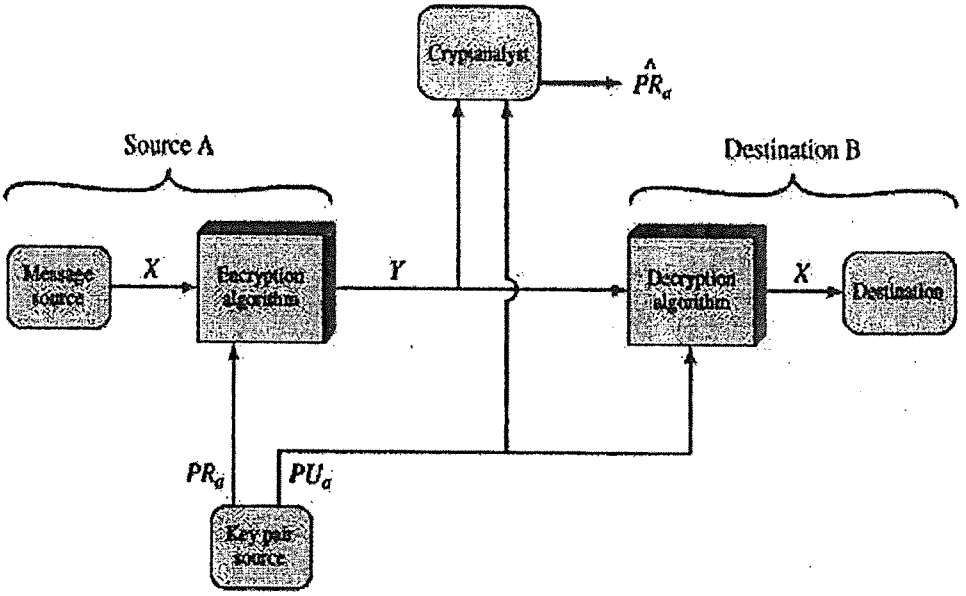
1. RotWord performs a one-byte circular left shift on a word. This means that an input word [B_0, B_1, B_2, B_3] is transformed into [B_1, B_2, B_3, B_0] .
2. SubWord performs a byte substitution on each byte of its input word, using the S-box
3. The result of steps 1 and 2 is XORed with a round constant, Rcon[j].

Requirements for Public-Key Cryptography:

The crypto system is described in the following diagram:



Public-Key Cryptosystem: Secrecy



Public-Key Cryptosystem: Authentication

The cryptosystem depends on a cryptographic algorithm based on two related keys. Diffie and Hellman proposed this system without demonstrating that such algorithms exist. However, they did lay out the conditions that such algorithms must fulfill [DIFF76b]:

1. It is computationally easy for a party B to generate a pair (public key PU_b, private key PR_b).
2. It is computationally easy for a sender A, knowing the public key and the message to be encrypted, M, to generate the corresponding ciphertext:

$$C = E(PU_b, M)$$

3. It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:

$$M = D(PR_b, C) = D[PR_b, E(PU_b, M)]$$

4. It is computationally infeasible for an adversary, knowing the public key, PU_b, to determine the private key, PR_b.
5. It is computationally infeasible for an adversary, knowing the public key, PU_b, and a ciphertext, C, to recover the original message, M.

We can add a sixth requirement that, although useful, is not necessary for all public-key applications:

6. The two keys can be applied in either order:

$$M = D[PU_b, E(PR_b, M)] = D[PR_b, E(PU_b, M)]$$

Public-Key Applications:

The applications of public-Key are:

1. The uses of public-key can classify into 3 categories:
   1.1. **encryption/decryption** (provide secrecy)
   1.2. **digital signatures** (provide authentication)
   1.3. **key exchange** (of session keys)
2. some algorithms are suitable for all uses, others are specific to one

RSA Algorithm:

The RSA was developed in 1977 by Ron Rivest, Adi Shamir, Len Adleman at MIT. Since then, the Rivest-Shamir-Adleman (RSA) scheme has become the most widely accepted and implemented general-purpose approach to public-key encryption.

This algorithm is used to encrypt integer data. In the RSA algorithm the integer message 'M' is encrypted by using the following equation:

$$\mathbf{C = M^e \bmod n}$$

The receiver uses the following equation for decryption:

$$\mathbf{M = C^d \bmod n}$$

Here both sender and receiver must know the integer value 'n'. The **sender** uses the public key $K_U = \{ e, n \}$. The **receiver** uses the private key $K_R = \{d, n\}$.

The following are the requirements for the RSA algorithm:

a) It is possible to find the values for e, d, n such that

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

b) It is easy to calculate M^e, C^d for all M, C.

c) It is computationally infeasible to find 'd' even { e, n } are known.

The RSA algorithm is stated as follows:

Key Generation:

1) Select two prime numbers p, q.

2) Calculate 'n', $n = p * q$

3) Calculate $\phi(n) = (p - 1) * (q-1)$

4) Select an integer 'e' such that $\gcd(\phi(n), e) = 1$

5) Find an integer 'd' such that $de \equiv 1 \bmod \phi(n)$

6) Form the public key $K_U = \{ e, n \}$

7) Form the private key $K_R = \{ d, n \}$

Encryption:

1) Prepare the message 'M' (M is an integer value)

2) Calculate cipher text $C = M^e \bmod n$

Decryption:

1) Receive the cipher text 'C'

2) Calculate plain text $M = C^d \bmod n$

In the RSA scheme, the following conditions are defined:

1) Select two prime numbers p and q (Selected, Private)

2) Calculate n (Calculated, Public)

3) Select 'e' such that $\gcd(\phi(n), e) = 1$ (Select, Public)

4) Calculate 'd' such that $de \equiv 1 \bmod \phi(n)$ (Calculated, Private)

5) K_U is public ($K_U = \{ e, n \}$)

6) K_R is private ($K_R = \{ d, n \}$)

Ex: 1

1) Key Generation:

i) Select two prime numbers p, q. Let $p = 7$ and $q = 17$

ii) Calculate $n = 7 * 17 = 119$

iii) Calculate $\phi(n) = 6 * 16 = 96$

iv) Select integer 'e' such that $\gcd(\phi(n), e) = 1 \Rightarrow \gcd(96, e) = 1$ and let $e = 5$

v) Calculate 'd' such that $de \equiv 1 \bmod \phi(n)$

$\Rightarrow de \bmod \phi(n) = 1$

$\Rightarrow d5 \bmod 96 = 1$

$\Rightarrow 5 * 77 \bmod 96 = 1$

$\Rightarrow d = 77$

vi) Form public key $K_U = \{5, 119\}$

vii) Form private key $K_R = \{77, 119\}$

Assume that M = 19

(Sender) Now $C = M^e \bmod n = 19^5 \bmod 119 = 66$

Now the receiver performs decryption in the following way:

$$M = C^d \bmod n = 66^{77} \bmod 119 = 19$$

Calculation of 19^5 mod 119:

$19^2 \bmod 119 \Rightarrow 361 \bmod 119 = 4$

$19^5 = (19^2 * 19^2 * 19) = (4 * 4 * 19) \bmod 119 = 304 \bmod 119 = 66$

Calculation of 66^{77} mod 119:

$66^2 \bmod 119 = 4356 \bmod 119 = 72$

$66^4 \bmod 119 = (66^2 * 66^2) \bmod 119 = (72 * 72) \bmod 119 = 67$

$66^8 \bmod 119 = (66^4 * 66^4) \bmod 119 = (67 * 67) \bmod 119 = 86$

$66^{16} \bmod 119 = (66^8 * 66^8) \bmod 119 = (86 * 86) \bmod 119 = 18$

$66^{32} \bmod 119 = (66^{16} * 66^{16}) \bmod 119 = (18 * 18) \bmod 119 = 86$

$66^{64} \bmod 119 = (66^{32} * 66^{32}) \bmod 119 = (86 * 86) \bmod 119 = 18$

$66^{77} \bmod 119 = (66^{64} * 66^8 * 66^4 * 66) \bmod 119$

$\Rightarrow (18 * 86 * 67 * 66) \bmod 119 = 19$

Ex: 2

Perform the encryption and decryption using RSA for the following data:

$p = 5, q = 11, e = 3, m = 9$

$n = 5 * 11 = 55$

$\phi(n) = 4 * 10 = 40$

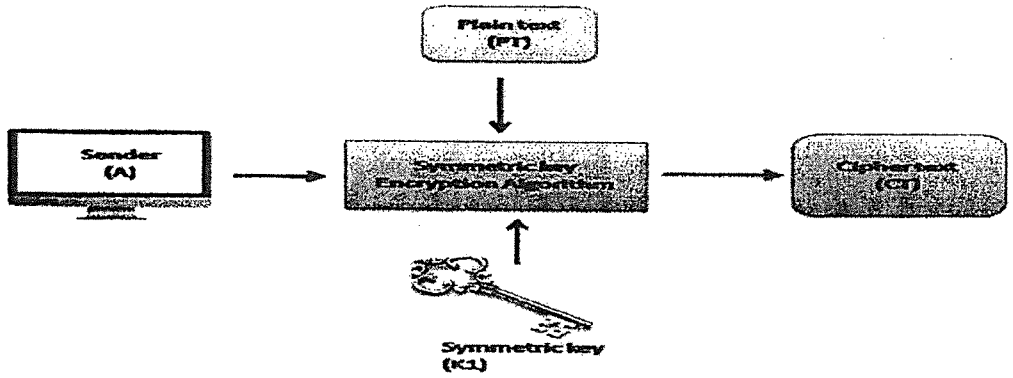
Select an integer 'e' such that $\gcd(\phi(n), e) = 1 \Rightarrow \gcd(40, 3) = 1$

Calculate 'd' such that $de \equiv 1 \bmod \phi(n)$

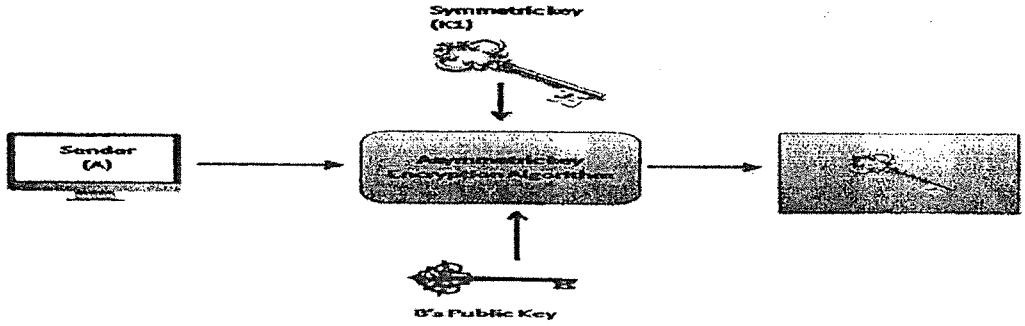
4. The solution should scale to a large number of users easily, without introducing any additional complications.
5. The key distribution problem must be solved by the solution.

Indeed, in practice, symmetric-key cryptography and asymmetric-key cryptography are combined to have a very efficient security solution. The way it works is as follows, assuming that A is the sender of message and B is its receiver.

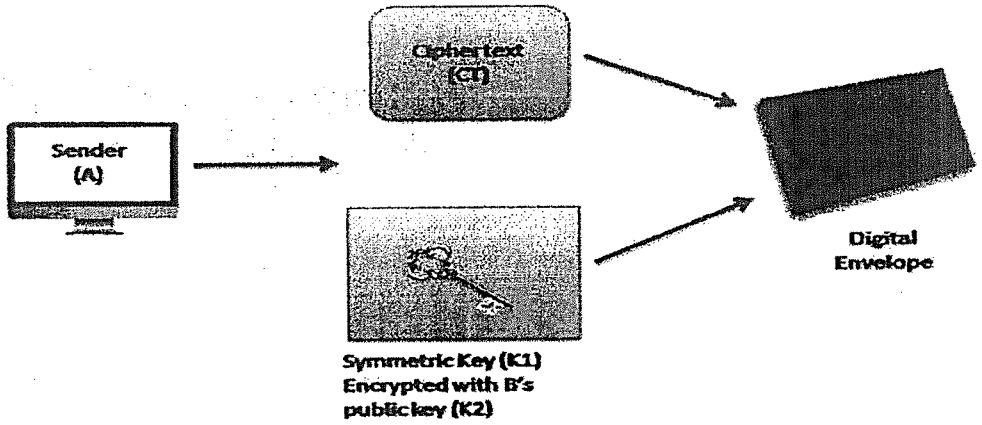
1. A's computer encrypts the original plain-text message (PT) with the help of a standard symmetric key cryptography algorithm, such DES, IDEA or RC5, etc. this produces a cipher-text message (CT) as shown in Fig. below. The key used in this operation (K_1) is called one-time symmetric key, as it is used once and then discarded.



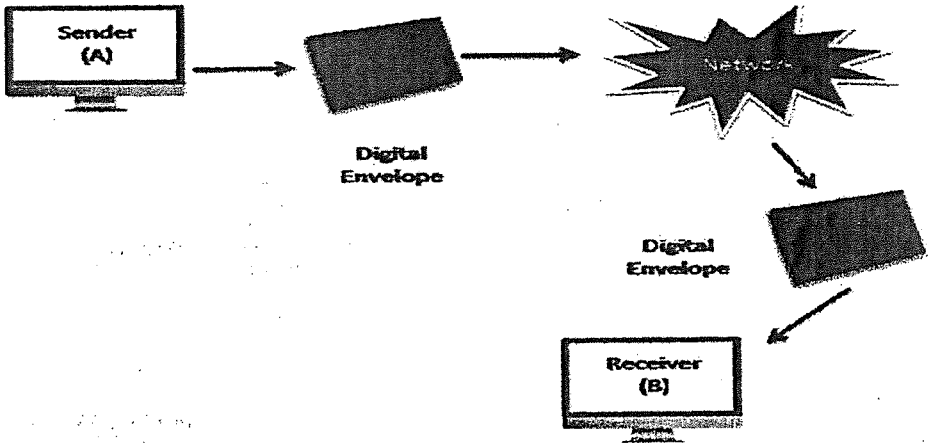
2. We would now think, we are back to square one! We have encrypted the plain text (PT) with a symmetric-key operation. We must now transport this one-time symmetric key (K_1) to the server so that the server can decrypt the cipher text (CT) to get back the original plain-text message (PT). Does this not again lead us to the key-exchange problem? Well, a novel concept is used now. A now takes the one-time symmetric key of step1 (i.e. K_1), and encrypts K_1 with B's public key (K_2). This process is called key wrapping of the symmetric key, and is shown in fig. below. We have shown that the symmetric key K1 goes inside a logical box, which is sealed by B's public key (i.e. K_2).



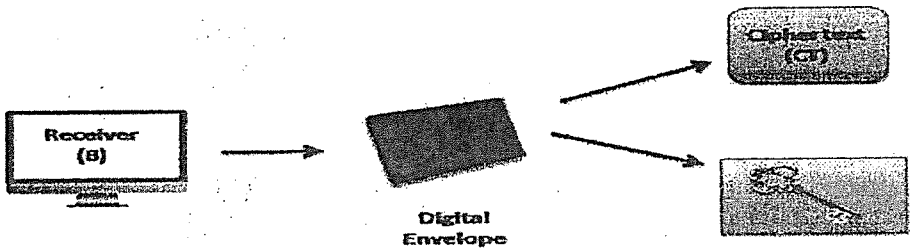
3. Now, A puts the cipher text CT1 and the encrypted symmetric key together inside a digital envelope. This is shown in the following figure:



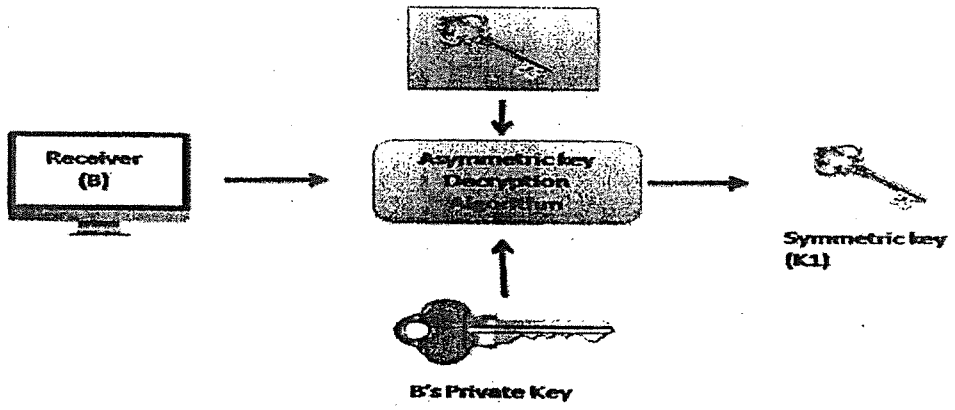
4. The sender (A) now sends the digital envelope [which contains the cipher text (CT) and the onetime symmetric key (K_1) encrypted with B's public key, (K_2)] to B using the underlying transport mechanism (network). This is shown in fig .we do not show the contents of the envelope, and assume that the envelope contains the two entities, as discussed.



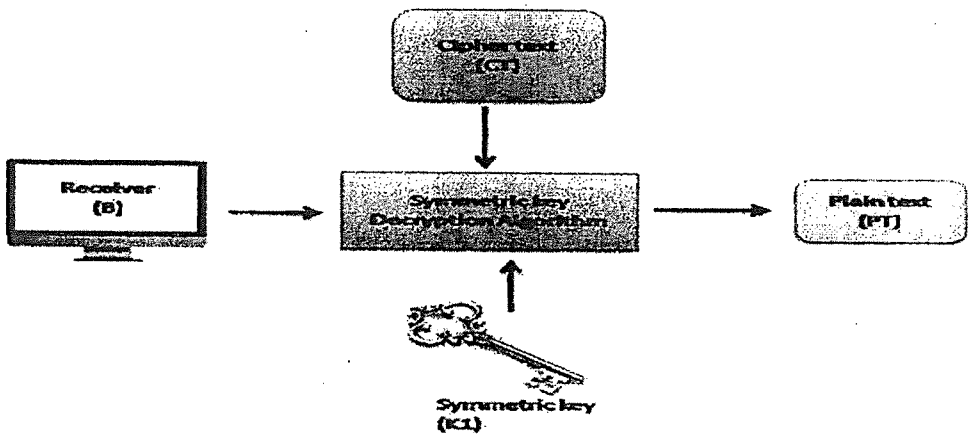
5. B receives digital envelope and opens it. After B opens this digital envelope, he gets 2 things first is cipher text (CT) and another one is the one-time session key (K_1) which is encrypted using B's public key (K_2). This is shown in the following figure:



6. B now uses the same asymmetric-key algorithm as was used by A and her private key (K_3) to decrypt (i.e. open up) the logical box that contains the symmetric key (K_1), which was encrypted with B's public key (K_2). This is shown in fig. below. This output of the process is the one-time symmetric key K_1.



7. Finally, B applies the same symmetric-key algorithm as was used by A, and the symmetric key K_1 to decrypt the cipher text (C1). This process yields the original plain text (PT), as shown in the following figure:



Authentication Requirements (Security Requirements):

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

1. **Disclosure:** Release of message contents to any person or process not possessing the appropriate cryptographic key.
2. **Traffic Analysis:** Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined.

In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.

3. **Masquerade:** Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or non-receipt by someone other than the message recipient.
4. **Content Modification:** Changes to the contents of a message, including insertion, deletion, transposition, and modification.
5. **Sequence Modification:** Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
6. **Timing Modification:** Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.
7. **Source Repudiation:** Denial of transmission of message by source.
8. **Destination Repudiation:** Denial of receipt of message by destination.

Authentication Functions:

There are three authentication functions. They are

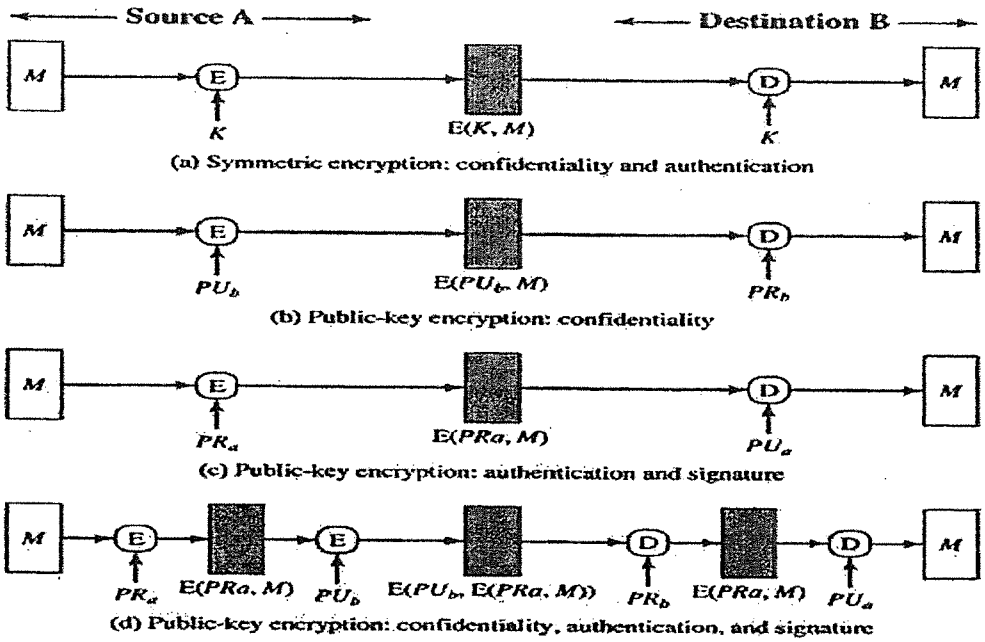
1. **Message encryption:** The ciphertext of the entire message serves as its authenticator
2. **Message authentication code (MAC):** A function of the message and a secret key that produces a fixed-length value that serves as the authenticator
3. **Hash function:** A function that maps a message of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

Message Encryption:

Message encryption by itself can provide a measure of authentication. There is a difference between for symmetric and public-key encryption schemes.

Symmetric Encryption:

A message M transmitted from source A to destination B is encrypted using a secret key K shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message. This is explained in the following diagram.



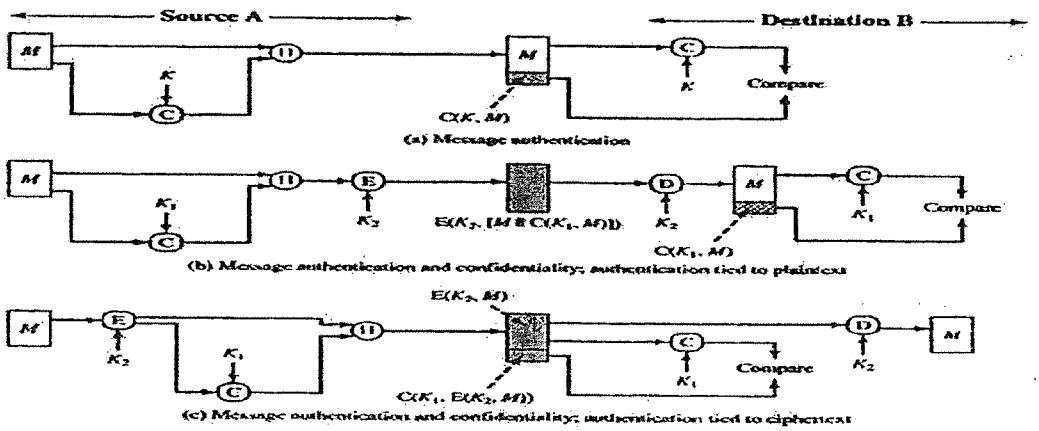
Basic Uses of Message Encryption

Public-Key Encryption:

This provides no confidence of sender because anyone potentially knows public-key. However, if sender **signs** message using their private-key, then encrypts with recipients public key. Hence it provides both secrecy and authentication.

Message Authentication Code (MAC):

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

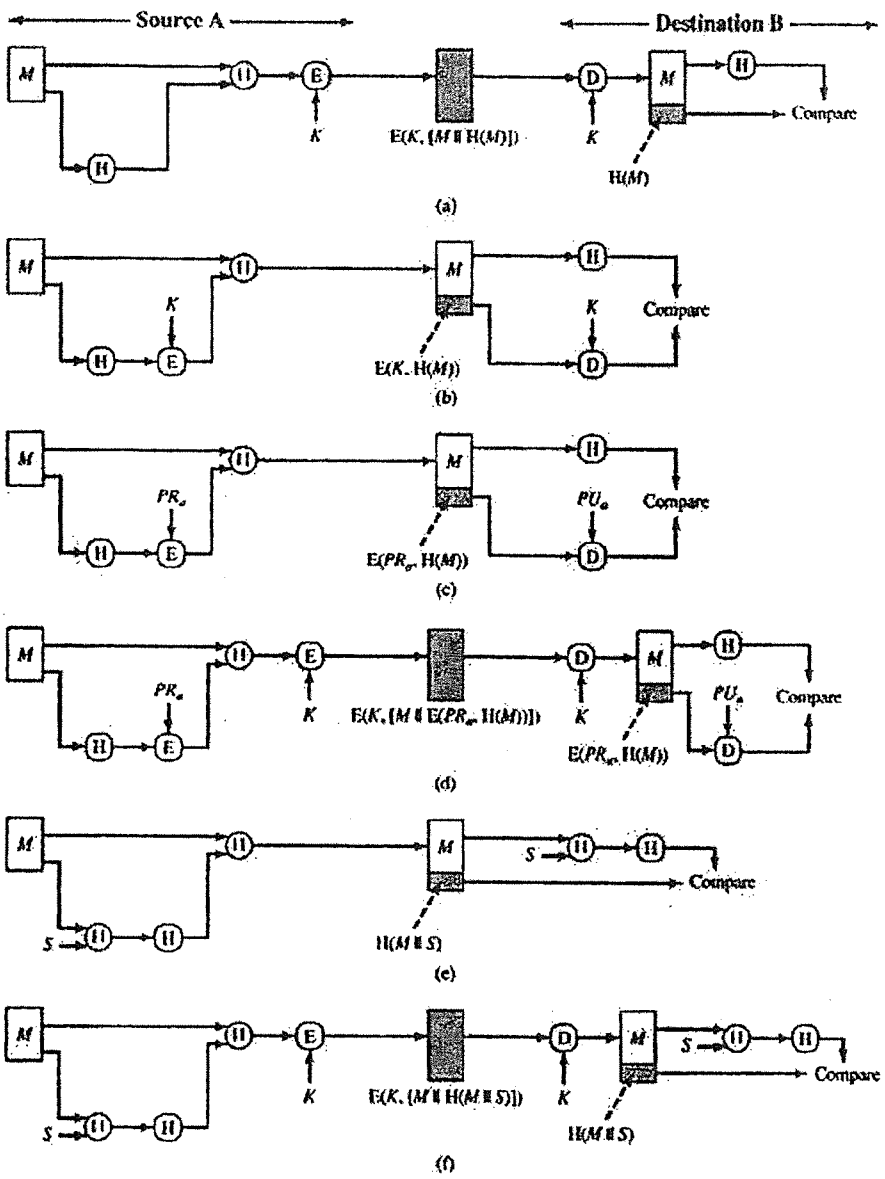


Basic Uses of Message Authentication Code (MAC)

Hash Function:

The hash algorithms are known as Message Digest Functions or One Way Transformation Functions. It is considered as a *function* because it takes an input message and produces an output. A hash function is a mathematical transformation, which takes arbitrary (random) length of message and computes a fixed length of numerical value. We call the hash of the message as H (M). We will use the terms *hash* and *message digest* interchangeably. The basic uses of hash functions are explained in the following table and diagram:

(a) $A \rightarrow B: E_K[M \mid H(M)]$ •Provides confidentiality —Only A and B share K •Provides authentication —H(M) is cryptographically protected	(d) $A \rightarrow B: E_K[M \mid E_{KRa}[H(M)]]$ •Provides authentication and digital signature •Provides confidentiality —Only A and B share K
(b) $A \rightarrow B: M \mid E_K[H(M)]$ •Provides authentication —H(M) is cryptographically protected	(e) $A \rightarrow B: M \mid H(M \mid S)$ •Provides authentication —Only A and B share S
(c) $A \rightarrow B: M \mid E_{KRa}[H(M)]$ •Provides authentication and digital signature —H(M) is cryptographically protected —Only A could create $E_{KRa}[H(M)]$	(f) $A \rightarrow B: E_K[M \mid H(M) \mid S]$ •Provides authentication —Only A and B share S •Provides confidentiality —Only A and B share K



Basic Uses of Hash Function

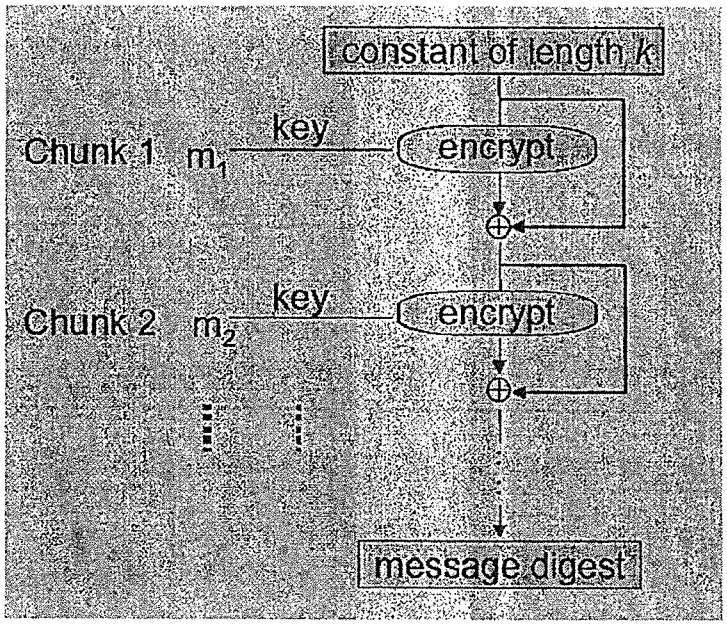
Birthday Problem:

Given a group of people, what is the minimum number of people such that two will share the same birthday with probability > 0.5?

1. For a single person P1, there are $n = 365$ possible birthdays → The probability $p1$ of having birthday at one of the days is n/n
2. For the second person P2, there are 364 (i.e. n-1) possibilities to have birthday at a day different from P1; the probability for this is p2 = (n / n) * (n – 1) / n
3. Generalizing - the probability of having birthday on different days for r people: Pr = n! / ((n – r)! * n^r)

4. The probability of a match is $1 - p_r$
5. On the average, a match will occur after $\approx n$ steps

How to compute a Message Digest?



1. *First idea*: convert a secret key algorithm into a message digest algorithm for arbitrary messages
2. Used e.g. to store hashes of UNIX passwords instead of the passwords it selves
3. *Given*: A secret key algorithm with key bits and message block length b bits (e.g. DES: k=56 and b=64)

Algorithm:

Split message m into k-bit chunks m_1, m_2, ...

Use m_1 as a key to encrypt a "constant"

Use m_2 to encrypt the previous result

.....

Use the final b-bit result as message digest

Problem: 64 bit message digest is too short

Generate a second 64-bit quantity using the chunks m_1, m_2, ... e.g. in reverse order.

MD5 (Message Digest 5) Algorithm:

This algorithm was developed by Ron Rivest. MD5 is seen as more secure and speed than MD4. This is a hash algorithm. This algorithm takes an arbitrary length of input message and produces a message digest of 128 bits. The input is processed as 512 bit blocks. This algorithm contains the following steps:

a) Append Padding Bits: This algorithm takes an arbitrary length of message. This message is padded, so that its length is congruent to 448 mod 512 i.e., length ≡ 448 mod 512, we can the bits as 1 followed by zeros.

Ex:

1. If the length of message contains 550 bits, we have to pad 410 bits i.e., 1 followed by 409 zeros to satisfy length mod 512 = 448.
2. If the original message length is 2000, we have to pad 496 bits.
3. If the original message length is 448, 960 or 1472, we have to pad 512 bits. The padding bits are always appended to the right of the original message.

b) Appending Length: Find out the length of the original message and represent it is 64 bits and are appended to the right side of the step (a) output. Now the total message consists 512 bit blocks. The output of step (b) contains a multiple of 512 bits.

c) Initializing MD Buffer: The MD5 algorithm uses a buffer of 128 bits. The buffer can be represented as 4 registers d_0, d_1, d_2 and d_3. The registers are initialized to the following Hexadecimal values.

d_0 = 67452301 (MSB (Most Significant Byte) = 67 & LSB (Least SB) = 01)

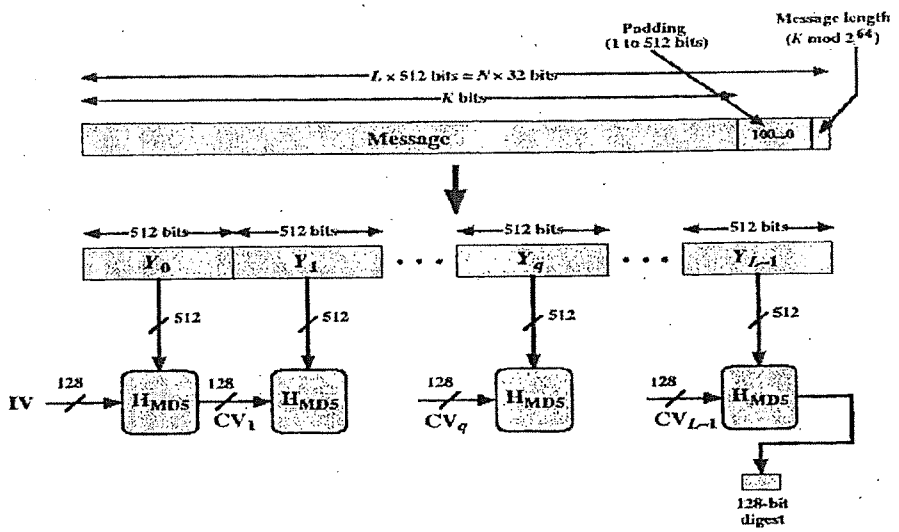
d_1 = efcdab89 (MSB (Most Significant Byte) = ef & LSB (Least SB) = 89)

d_2 = 98badcfe (MSB (Most Significant Byte) = 98 & LSB (Least SB) = fe)

d_3 = 10325476 (MSB (Most Significant Byte) = 67 & LSB (Least SB) = 01)

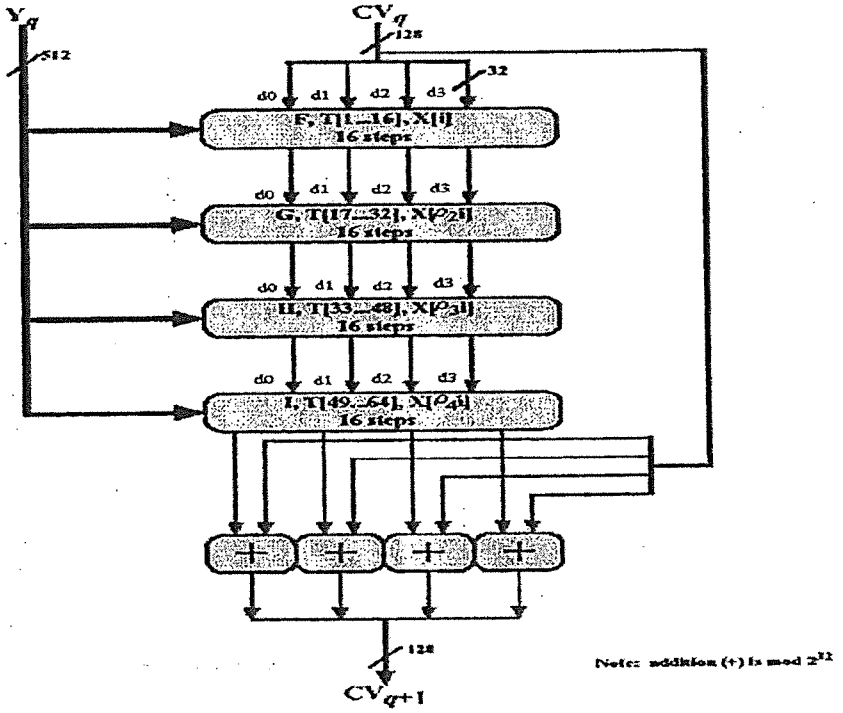
Now these values are stored in little Endean format i.e., the least significant byte will store in low address. Now d_0 = 01234567 d_1 = 89abcdef d_2 = fedcba98 d_3 = 76543210

d) We divide the entire message (original message + appending padding + appending length) into 512 bit blocks and for each block a separate HMD5 is applied. This is shown in the following diagram (IV=Initial Value for the d_0, d_1, d_2 and d_3 buffer (4 * 32-bits=128–bits).



Processing of the 512 bit blocks:

The heart of the algorithm is called a compression function. It consists 4 rounds. The 4 rounds are having similar structure, but each round uses a different primitive function. The following diagram explains this concept.



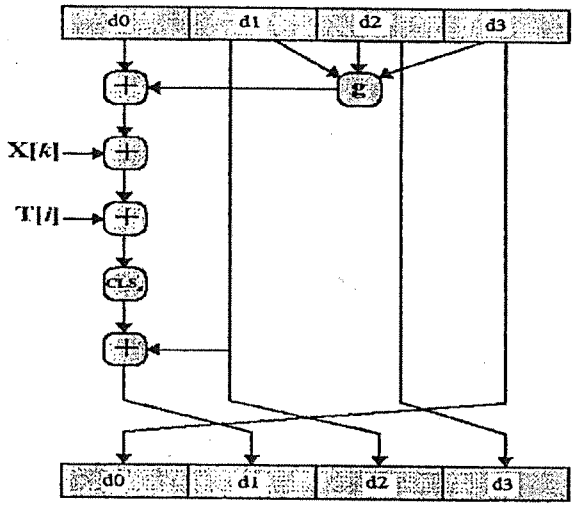
CV_q = Changing Variable processed with q^{th} block of the message

Y_q = The q^{th} 512 – bit block of the message

From the above diagram, the following expressions are used in the MD5 algorithm.

MD5 Compression Function:

In MD5 algorithm, we have 4 rounds and each round contains 16 steps in it. These steps operate on the d_0, d_1, d_2 and d_3. The following diagram explains the internal structure of each step:



From the above compression function, the following functions are used.

$d_0 = d_3$ **CLS = Circular Left Shift**

$d_1 = sum_{32}\ (\ d_1, CLS\ (\ T_i\ (\ X_k, (\ d_0, g\ (d_1, d_2, d_3\ )\ )\ )\ )\ )$

$d_2 = d_1$ **sum_{32} = Addition of modulo 2^{32} performed separately on each word of the pair of inputs**

$d_3 = d_2$ **g = One of the primitive functions of F, G, H, I**

The following primitive functions are used in HMD5 algorithm.

Pass 1: Selection function

• Calculation bases on function $F(x, y, z) = (x \wedge y) \vee (\sim x \wedge z)$

'i' value varies from 0 through 15.

$$d_{(-i)\wedge 3} = (d_{(-i)\wedge 3} + F(d_{(1-i)\wedge 3}, d_{(2-i)\wedge 3}, d_{(3-i)\wedge 3}) + m_i + T_{i+1}) \hookleftarrow S_1(i \wedge 3)$$

where $S_1 = 7 + 5i$:

$$\begin{aligned} d_0 &= (d_0 + F(d_1, d_2, d_3) + m_0 + T_1) \hookleftarrow 7 \\ d_3 &= (d_3 + F(d_0, d_1, d_2) + m_1 + T_2) \hookleftarrow 12 \\ d_2 &= (d_2 + F(d_3, d_0, d_1) + m_2 + T_3) \hookleftarrow 17 \\ d_1 &= (d_1 + F(d_2, d_3, d_0) + m_3 + T_4) \hookleftarrow 22 \\ d_0 &= (d_0 + F(d_1, d_2, d_3) + m_4 + T_5) \hookleftarrow 7 \end{aligned}$$

Pass 2: Second selection function

• Calculation bases on function $G(x, y, z) = (x \wedge z) \vee (y \wedge \sim z)$

$$d_{(-i)\wedge 3} = (d_{(-i)\wedge 3} + G(d_{(1-i)\wedge 3}, d_{(2-i)\wedge 3}, d_{(3-i)\wedge 3}) + m_{(5i+1)\wedge 15} + T_{i+17}) \hookleftarrow S_2(i \wedge 3)$$

where $S_2(i) = i(i+7)/2 + 5$:

$$\begin{aligned} d_0 &= (d_0 + G(d_1, d_2, d_3) + m_1 + T_{17}) \hookleftarrow 5 \\ d_3 &= (d_3 + G(d_0, d_1, d_2) + m_6 + T_{18}) \hookleftarrow 9 \\ d_2 &= (d_2 + G(d_3, d_0, d_1) + m_{11} + T_{19}) \hookleftarrow 14 \\ d_1 &= (d_1 + G(d_2, d_3, d_0) + m_0 + T_{20}) \hookleftarrow 20 \\ d_0 &= (d_0 + G(d_1, d_2, d_3) + m_5 + T_{21}) \hookleftarrow 5 \end{aligned}$$

Pass 3: First modification function

• Calculation bases on function $H(x, y, z) = x \oplus y \oplus z$

$$d_{(-i)\wedge 3} = (d_{(-i)\wedge 3} + H(d_{(1-i)\wedge 3}, d_{(2-i)\wedge 3}, d_{(3-i)\wedge 3}) + m_{(3i+5)\wedge 15}) + T_{i+33}) \hookleftarrow S_3(i \wedge 3)$$

where $S_3(i) = 4, 11, 16, 23, 4, \ldots$

$$\begin{aligned} d_0 &= (d_0 + H(d_1, d_2, d_3) + m_5 + T_{33}) \hookleftarrow 4 \\ d_3 &= (d_3 + H(d_0, d_1, d_2) + m_8 + T_{34}) \hookleftarrow 11 \\ d_2 &= (d_2 + H(d_3, d_0, d_1) + m_{11} + T_{35}) \hookleftarrow 16 \\ d_1 &= (d_1 + H(d_2, d_3, d_0) + m_{14} + T_{36}) \hookleftarrow 23 \\ d_0 &= (d_0 + H(d_1, d_2, d_3) + m_1 + T_{37}) \hookleftarrow 4 \end{aligned}$$

Pass 4: Second modification function

• Calculation bases on function $I(x, y, z) = y \oplus (x \vee \sim z)$

$$d_{(-i)\wedge 3} = (d_{(-i)\wedge 3} + I(d_{(1-i)\wedge 3}, d_{(2-i)\wedge 3}, d_{(3-i)\wedge 3}) + m_{(7i)\wedge 15} + T_{i+49}) \hookleftarrow S_4(i \wedge 3)$$

where $S_4(i) = 6, 10, 15, 21, 6, \ldots$

$$\begin{aligned} d_0 &= (d_0 + I(d_1, d_2, d_3) + m_0 + T_{49}) \hookleftarrow 6 \\ d_3 &= (d_3 + I(d_0, d_1, d_2) + m_7 + T_{50}) \hookleftarrow 10 \\ d_2 &= (d_2 + I(d_3, d_0, d_1) + m_{14} + T_{51}) \hookleftarrow 15 \\ d_1 &= (d_1 + I(d_2, d_3, d_0) + m_5 + T_{52}) \hookleftarrow 21 \\ d_0 &= (d_0 + I(d_1, d_2, d_3) + m_{12} + T_{53}) \hookleftarrow 6 \end{aligned}$$

In the MD5 algorithm, we add a T_i value in each step. The T_i value is found in the following way. It is equal to an integer part of 2^{32} * abs (sin (i)).

E.g. : T [1] = Integer part of 2^{32} * abs (sin (1))

= D7A6A478

In MD5 algorithm we use X_i values. The X_i values are calculated in the following way. First, we have to consider the 512 bits input to the round. It divided into 16 equal parts

and they are called X [1] to X [16]. These are used in first 16 steps or in pass 1. For the remaining passes (2, 3, 4), we found the X values in the following way.

$\rho_2(i) = (1 + 5i) \bmod 16$

$\rho_3(i) = (5 + 3i) \bmod 16$

$\rho_4(i) = 7i \bmod 16$

In this way, the MD5 compression function will be worked.

Differences between MD4 and MD5:

MD5 is speed and security than MD4. MD5 is similar to MD4. The major differences are:

a. MD4 takes three passes over each 16-octet chunk of the message. MD5 makes four passes over each 16-octet chunk.

b. The functions are slightly different, as are the number of bits in the shifts.

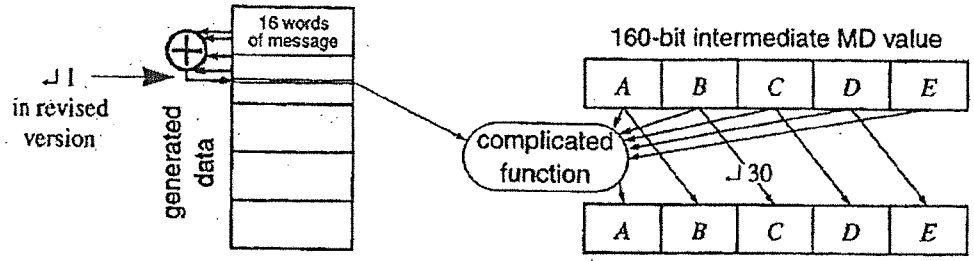
c. MD4 has one constant which is used for each message word in pass 2and a different constant used for all the 16 message words in pass 3. No constant is used in pass 1.

d. MD5 uses different constants for each message word on each pass. Since there are 4 passes, each of which deals with 16 message words, there are 64 32-bit constants used in MD5. We call them T_1 through T_{64}. T_i is based on the sine function, i.e., $T_i = [\ 2^{32} \mid \sin i \mid]$ = Integer part of 2^{32} * abs (sin (i))

Strength of MD5:

1. MD5 hash is dependent on all message bits
2. Rivest claims security is good as can be
3. known attacks are:
    a. Berson 92 attacked any 1 round using differential cryptanalysis (but can't extend)
    b. Boer & Bosselaers 93 found a pseudo collision (again unable to extend)
    c. Dobbertin 96 created collisions on MD compression function (but initial constants prevent exploit)
4. conclusion is that MD5 looks vulnerable soon

SHA – 1 (Secure Hash Algorithm):



Inner Loop of SHA-1—80 Iterations per Block

The Secure Hash Algorithm takes an arbitrary length of input message and produces a 160-bit message digest value.

The ***algorithm*** proceeds in the following way:

1. Take message and add padding (1 followed by 0's) so that the entire message length, including the padding, is evenly divided by 512.
2. The SHA algorithm works on blocks of 512.
3. Establish W_n for processing such that $0 <= n <= 79$
   a) W_n is a length of 32 bits
   b) Segment the 512 bit block into 16 blocks of 32 bits to obtain W0 through W15
   c) Starting with n=16, use the following calculation to obtain W16 through W79: $W_n = W_{n-3} \oplus W_{n-8} \oplus W_{n-14} \oplus W_{n-16}$
4. The SHA-1 uses a 160 bit buffer. It is divided into *Five* 32 bit registers called A, B, C, D and E. The initialized values of A, B, C, D, and E are as follows:

   A = 67452301 B = EFCDAB89 C = 98BADCFE D = 10325476

   E = C3D2E1F0
5. For each round from 0 through 79 A, B, C, D, and E are processed as follows:

   A = E + (A 5) + W_t + K_t + f(t, B, C, D)

   B = old A

   C = old B 30

   D = old C

   E = old D

Where W_t is the t^{th} 32 – bit word in the 80-word block. K_t is a constant.

a. Each round also uses the additive constants K_t, where $0 \leq t \leq 79$ indicates one of the 80 steps across 4 rounds

b. In fact only 4 constants are used:

Step Number	Hexadecimal	Inetger Part of
$0 \leq t \leq 19$	K_t = 5A827999	$[2^{30} \times \sqrt{2}]$
$20 \leq t \leq 39$	K_t = 6ED9EBA1	$[2^{30} \times \sqrt{3}]$
$40 \leq t \leq 59$	K_t = 8F1BBCDC	$[2^{30} \times \sqrt{5}]$
$60 \leq t \leq 79$	K_t = CA62C1D6	$[2^{30} \times \sqrt{10}]$

6. Each primitive function takes three 32-bit words as input and produces a 32-bit word output. Each function performs a set of bitwise logical operations

a) For 0 <= t <= 19, f(t, B, C, D) = (B ^ C) v (~B ^ D) and K_t = 5A827999

b) For 20 <= t <= 39, f(t, B, C, D) = B ⊕ C ⊕ D and K_t = 6ED9EBA1

c) For 40 <= t <= 59, f(t, B, C, D) = (B ^ C) v (B ^ D) v (C ^ D) and K_t = 8F1BBCDC

d) For 60 <= t <= 79, f(t, B, C, D) = B ⊕ C ⊕ D and K_t = CA62C1D6

Step	Function Name	Function Value
($0 \leq t \leq 19$)	$f_1 = f(t,B,C,D)$	$(B \wedge C) \vee (B' \wedge D)$
($20 \leq t \leq 39$)	$f_2 = f(t,B,C,D)$	$B \oplus C \oplus D$
($40 \leq t \leq 59$)	$f_3 = f(t,B,C,D)$	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
($60 \leq t \leq 79$)	$f_4 = f(t,B,C,D)$	$B \oplus C \oplus D$

Truth table

B	C	D	f_1	f_2	f_3	f_4
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	0	1	0	1
1	0	1	0	0	1	0
1	1	0	1	0	1	0
1	1	1	1	1	1	1

7. The result of the last round n=79 is added to the initial values of A, B, C, D, and E to obtain the 160 (32*5) bit message digest for the block.

Comparison of SHA-1 and MD5:

The two algorithms are compared using their design goals. They are

Security against brute-force attacks: The most important difference is that SHA-1 digest is 32-bits longer than MD5 digest.

1. **Security against cryptanalysis:** MD5 is vulnerable to cryptanalytic attacks, where as SHA-1 not be vulnerable to such attacks.
2. **Speed:** SHA-1 should execute more slowly than MD5 due to size of digest and number iterations.
3. **Simplicity and compactness:** Both algorithms are simple to describe and implement. They do not require larger programs and substitution tables.
4. **Little-endian versus big-endian architecture:** MD5 uses little-endian scheme for message sequence for 32-bit words. But SHA-1 uses big-endian scheme and is not giving strong advantage.

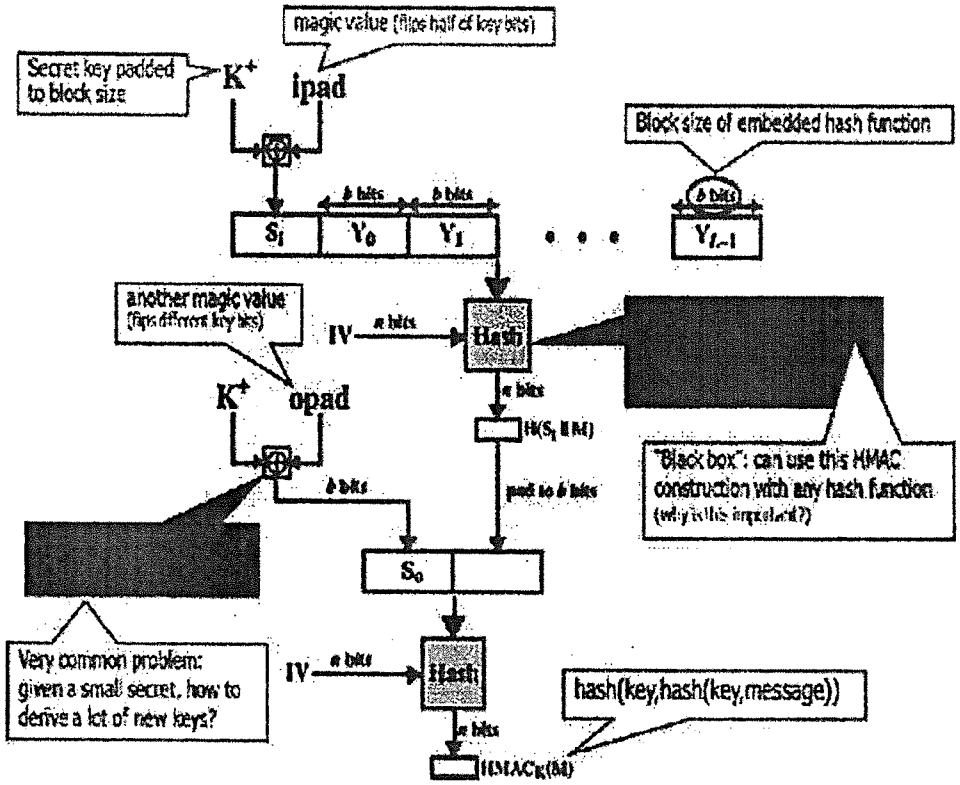
HMAC (MAC with Hash):

Hash algorithms like SHA -1 and MD5 could not use any secret key, whereas the MAC algorithm uses a secret key. If we combine these two, so that key is used in Hashing. Hence, this algorithm is known as HMAC algorithm.

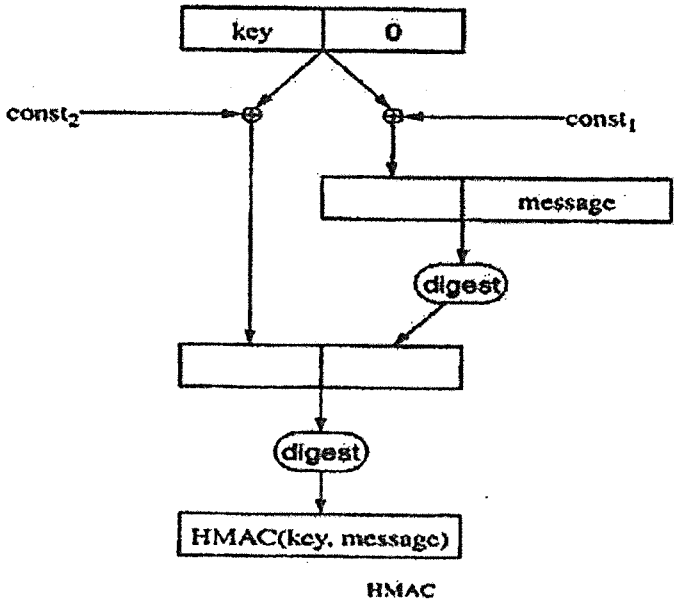
This ***algorithm*** consists the following steps:

1. Consider the arbitrary length message M and divide it in to 'L' 512 bit blocks. Each block length is denoted by 'b'.
2. Initially we generate a secret key of length 'n'. Now we pad some zeros on left of
3. 'k' to get m-bit key (Here $m < b$). Now the key is padded with zeros on left. So the length of key is equal to 'b'. We denote it as K^+.
4. Initialize ipad = 00110110 repeated b / 8 times. The length of ipad = 512 bits or b.
5. Initialize opad = 01011010 repeated b / 8 times. The length of opad = 512 bits or b.
6. Apply the *Exclusive-or* on K^+ and ipad to get a b-bit output block called S_i.
7. S_i is appended to the message blocks y_0 to y_{L-1} at the beginning. So, we have a total of L + 1 blocks.
8. Apply the Hash function to the above blocks to get n-bit Hash value H ($S_i \| M$)
9. Apply the *Exclusive or* operation on K^+ and opad to get S_o of b-bits.
10. S_0 is appended to the output of H in *step 7* at the beginning.
11. H ($S_i \| M$) is padded with zeros on left side until it becomes a b-bit blocks.
12. Now we apply H to the output of *step 10*. The result contains n-bits and this is called HMAC value of the given message.

The above all steps are shown in the following figure:



In a brief HMAC is explained in the following figure:



HMAC

HMAC Security:

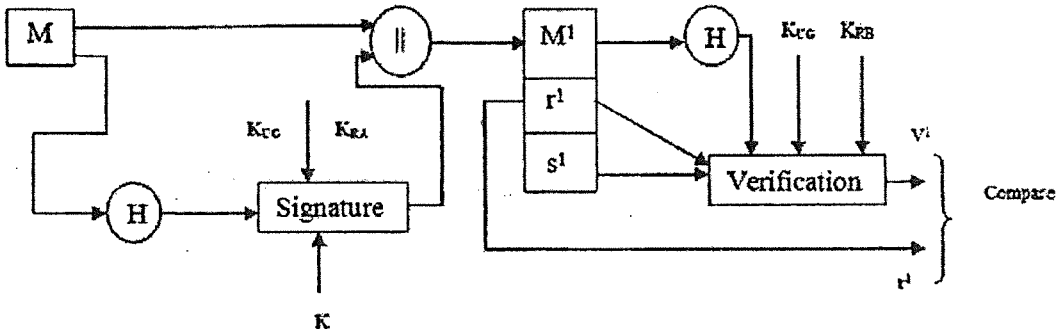
1. We know that the security of HMAC relates to that of the underlying hash algorithm
2. To attack HMAC requires either:
   a. brute force attack on key used
   b. birthday attack (but since keyed would need to observe a very large number of messages)

To choose hash function used based on speed verses security constraints

Digital Signature Standard (DSS) Algorithm:

This Algorithm is known as the Digital Signature Algorithm. This was developed by NIST, National Institute of Standards and Technology in 1991. This employs an irreversible public key system.

In the DSS approach sender prepares a message and calculates Hash value and sign on it. The signed Hash value is appended to the original message and sends it to the receiver. The receiver verifies the signature and if it matches, he accepts the message. The following diagram explains this idea: K_{RB}



Algorithm:

1. Select a large prime number p
2. Select prime number q which is a divisor of (p-1) where $2^{159}<q<2^{160}$
3. Select a number h where $1<h<(p-1)$
4. Calculate $g = h^{(p-1)/q} \bmod p$ such that $h^{(p-1)/q} \bmod p > 1$
5. Select a private key which is random number x where $0<x<q$
6. Then the public key is $y = g^x \bmod p$

Per Message Application:

1. Select a random integer k such that $0<k<q$
2. Let M be the message to be transmitted

3. Let H(M) be the hash of the message to be transmitted using SHA-1. SHA-1 closely models MD4
4. Calculate r (text uses Tm), where $r = (g^k \bmod p) \bmod q$
5. Calculate s (text uses X), where $s = (k^{-1}(H(M) + x*r) \bmod q$
6. Transmit M, r, and s.

Verifying Signature:

1. Calculate $(s^1)^{-1} \bmod q = w$
2. Calculate $H(M^1)$
3. Calculate u1 (text uses x), where $u1 = [H(M^1)*w] \bmod q$
4. Calculate u2 (text uses y), where $u2 = (r^1*w) \bmod q$
5. Calculate v, where $v = [(g^{u1} * y^{u2}) \bmod p] \bmod q$
6. If v = r then the signature is verified

Signature Function:

In the DSS algorithm, we use three global key elements. They are p (Prime Number), q (is divisor of p -1) and g ($h^{(p-1)/q} \bmod p$ such that $h^{(p-1)/q} \bmod p > 1$). These three values make K_{UG} (Global public Key Elements).

The signature function uses the secret key of sender i.e., private key $X = K_{Ra}$. The signature function uses a secret number 'k'. It is selected randomly. The sender prepares the message and calculates its Hash value and gives it to the signature function along with K_{UG}, K_{Ra} and k. Now the signature is produced. It contains two components r and s.

$r = (g^k \bmod p) \bmod q$ $\qquad$ $s = (k^{-1}(H(M) + x*r) \bmod q$

Signature Verification:

After receiving M^1, r^1, s^1, the receiver first calculates the hashed value of the received message. Then the receiver needs to collect K_{UG}, K_{Ua}. The K_{UG} contains three components p, q, g and the public key of sender is also collected. Earlier the sender calculates his public key using the following formulae:

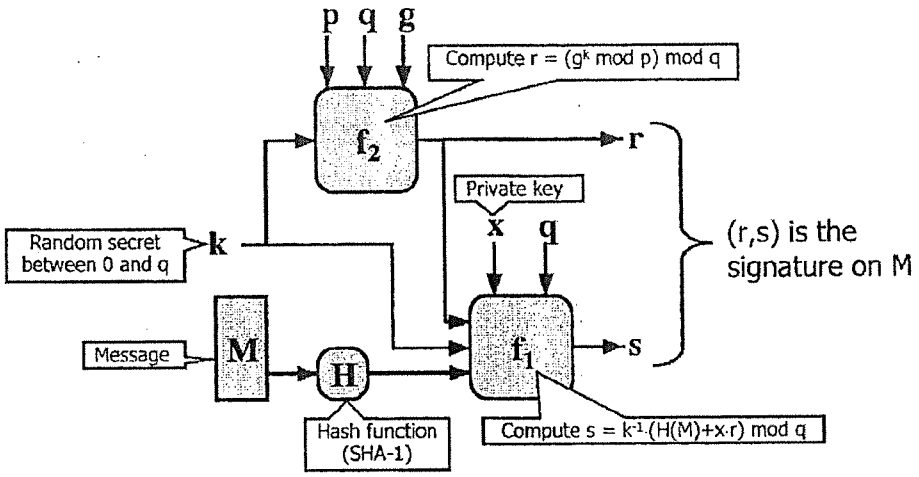
$K_{Ua} = y = g^x \bmod p$ and the public key is kept opened to all.

Now the receiver gives H (M1), r^1, s^1, K_{UG} and K_{UA} to verification function. The following expressions are used in verification function:

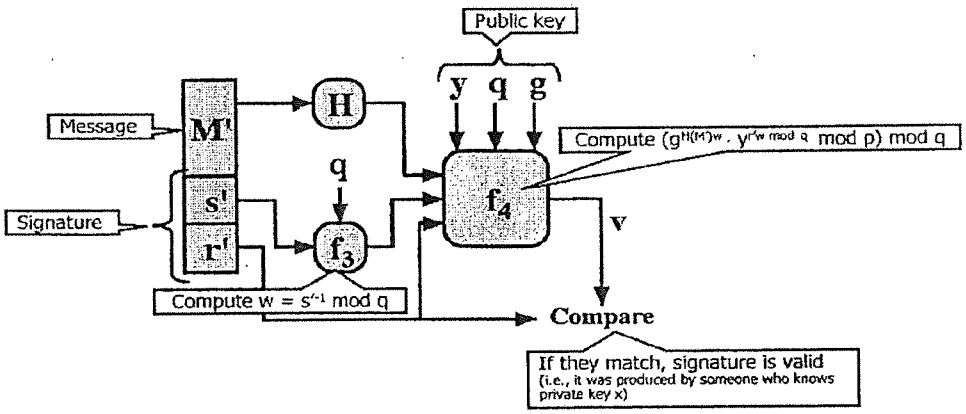
$w = (s^1)^{-1} \bmod q$ $\quad u_1 = [H(M^1)*w] \bmod q$ $\quad u_2 = (r^1*w) \bmod q$ $\quad v = [(g^{u1} * y^{u2}) \bmod p] \bmod q$

Now the receiver is going to test whether $v = r^1$ or not. If these two are equal means the signature is verified. Otherwise, signature is not matched. If signature is matched, receiver accepts M^1, otherwise, he rejects M^1. *The following diagram explains how signature is generated and verified:*

Signature = (r, s):



Verification:



****

UNIT – III

1. User Authentication Mechanisms: Introduction - Authentication basics - passwords authentication tokens - certificate based authentication - biometrics authentication - Hash functions - SHA1.

2. System Security: Intruders, Viruses, Related Threats, Trusted Systems.

1. User Authentication Mechanisms

Authentication:

" **Verifying the identity of another entity**"

– Computer authenticating to another computer

– Person authenticating to a local computer

– Person authenticating to a remote computer

What is Authorization?

The method that is used to determine the resources that are accessible to an authenticated user is called authorization. For example, in a database, set of users are allowed to update/ modify the database, while some users can only read the data.

What is the difference between Authentication and Authorization?

Authentication is the process of verifying the identity of a user who is trying to gain access to a system, whereas authorization is a method that is used to determine the recourses that are accessible to an authenticated user. Even though authentication and authorization performs two different tasks, they are closely related.

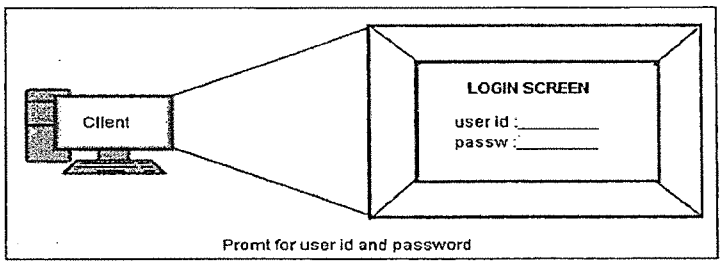
Passwords:

A password is a string of alphabets, numbers and special characters, which is supposed to be known only to the entity (usually a person) that is being authenticated.

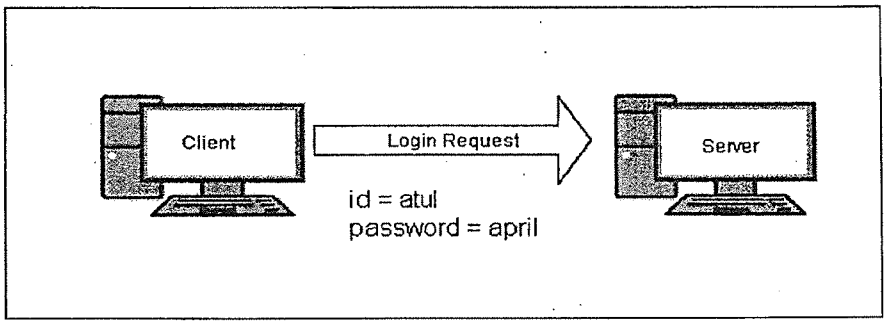
Clear Text Passwords:

This is the simplest password-based authentication mechanism. Usually, every user in the system is assigned a user id and an initial password. The user changes the password periodically for security reasons. The password is stored in clear text in the user database against the user id on the server. The authentication mechanism works as follows:

Step 1: Prompt for user id and password: During authentication the application sense a screen to the user, prompting for the user id and password. This is shown in the following figure:

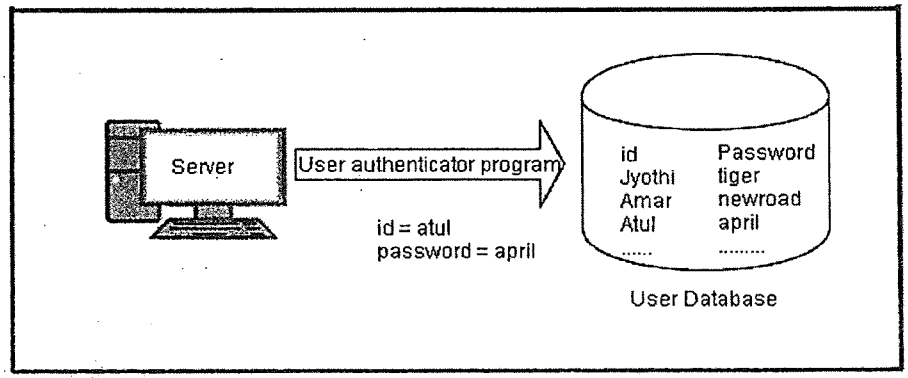


Step 2: User enters user id and password: The user enters her id and password and presses the OK equivalent button. This causes the user id and password to travel in clear text to the server. This is shown in the following figure:



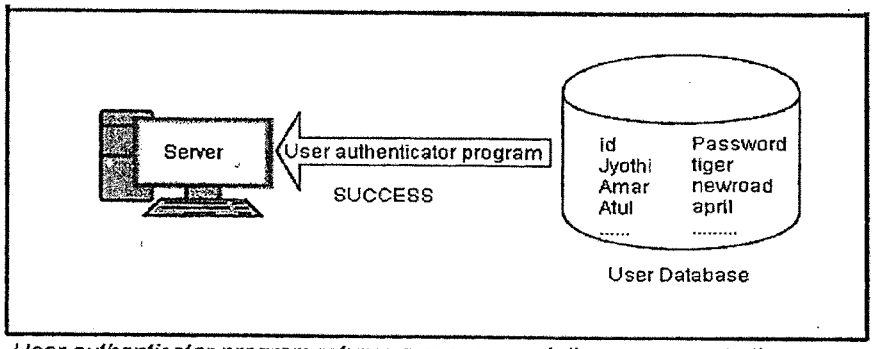
User id and password travel in clear text to the server

Step 3: User id and password validation: The server consults the user database to see if this particular user id and password combination exists there. Usually, this is the job of a user authenticator program is shown in the following figure. This is a program that takes user id and password, checks it against user database, and returns the result of the authentication (success or failure). This is one of the types of checking.



User authenticator checks the user id and password against the user database

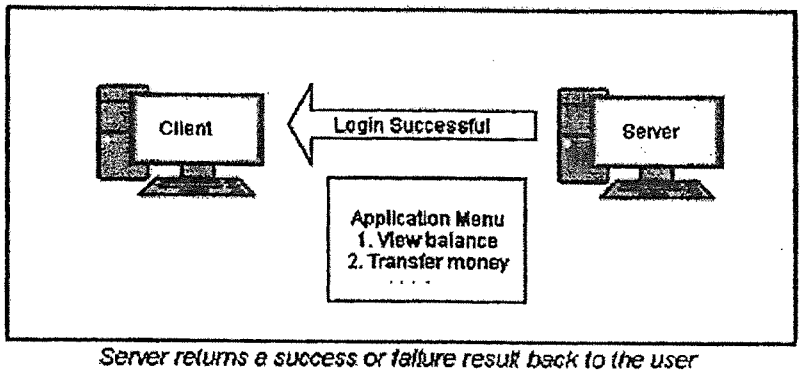
Step 4: Authentication Result: Depending on the success or the failure of the validation of the user id on the password, the user authenticator program returns an appropriate result back to the server which is shown in the following diagram.



User authenticator program returns a success or failure message to the server

Here, we assume that the user was authenticated successfully.

Step 5: Inform user accordingly: Depending on the outcome (success/failure), the server sense back an appropriate screen to the user. If the user authentication was successful, the server typically sends a menu of options of the user, which lists the actions the user is allowed to perform. If the result of the user authentication was a failure, the server sends an error screen to the user which is shown in the following figure.



Server returns a success or failure result back to the user

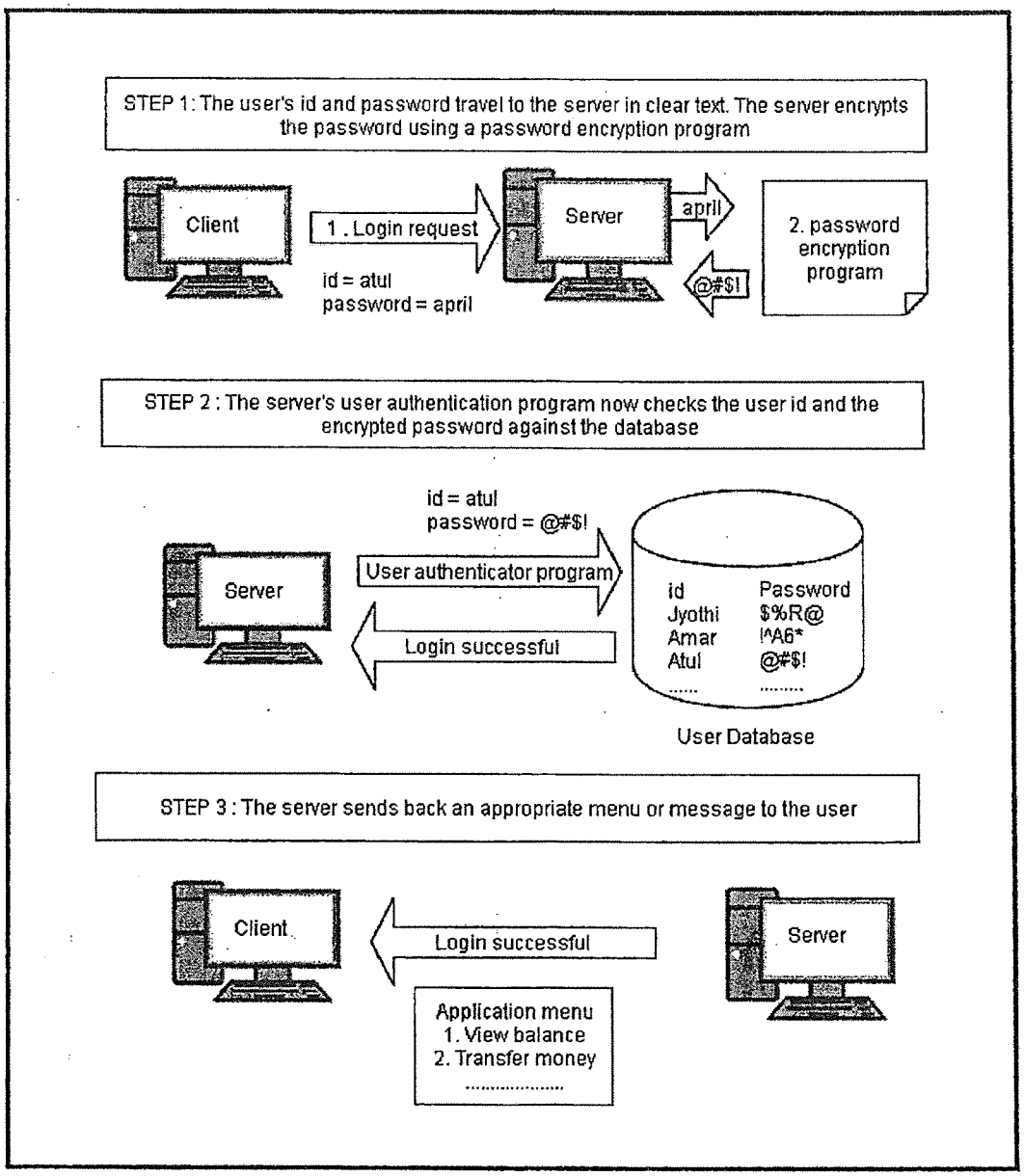
Here we assume that the user was authenticated successfully.

Problems with the scheme:

There are two major problems in this approach.

Problem 1: The database consists user ids and passwords in clear text

If an attacker succeeds to access the database, he can get whole list of user ids and corresponding passwords and also he can understand the passwords because they are in the clear text. The solution to this problem is the passwords should be encrypted. This is shown in the following:



Encryption password before they are stored and verified

Problem 2: Password travels in clear text from the user's computer to the server

Though the server stored passwords in the encryption format, the attacker attacks while the password is traveling from user's computer to the server for authentication and he can understand the password easily because the password in the clear text format.

Something derived from Passwords:

Instead of storing password as it is, the password is passed as input to the encryption algorithm and the output password is in the encrypted format which is stored in the server. When the user wants to be authenticated, the user enters password and the user's computer

performs same algorithm locally and sends the derived password to the server, where it is verified. But it requires several requirements need to be met to make this approach success. They are:

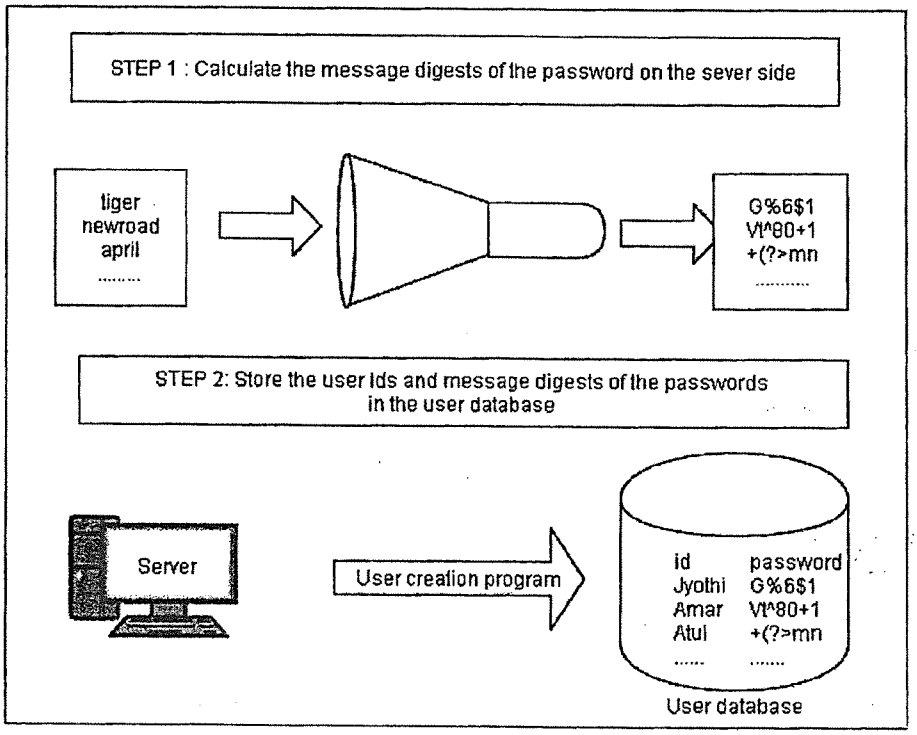
1. Each the algorithm is executed for the same password; it must produce the same output.
2. The output of the algorithm must not provide any clues regarding the original password.
3. It should be infeasible for an attacker to provide an incorrect password and yet obtain the correct derived password.

Message Digests of Passwords:

This is a simple technology to avoid the storage and transmission of clear text passwords. The following steps explain the working nature of this technique.

Step 1: Storing message digests as derived passwords in the user database

Rather than storing passwords, we can store the message digests of the passwords in the database is shown in the following figure:

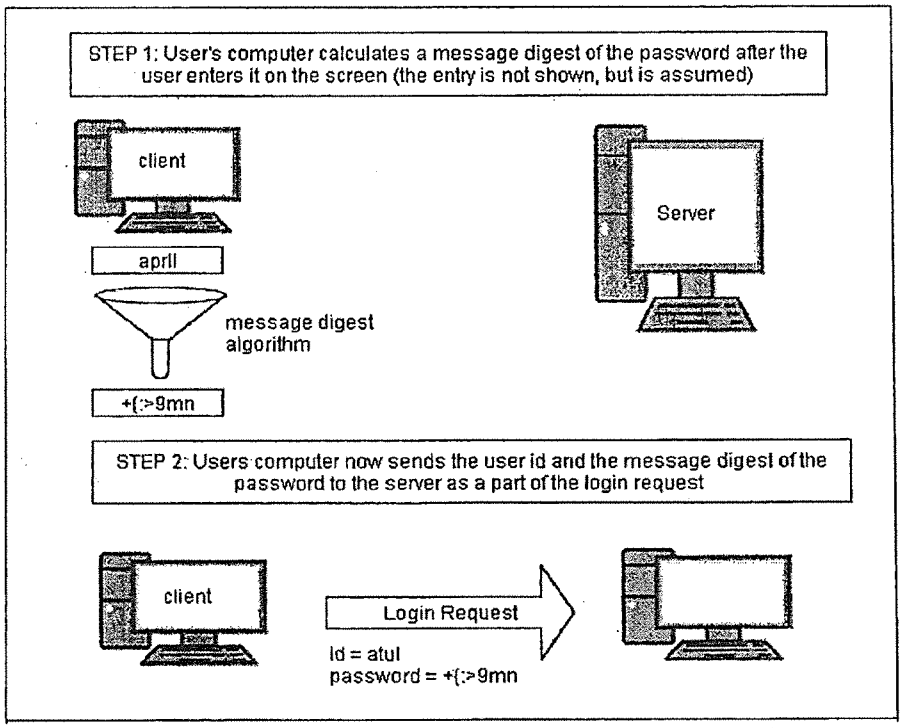


Storing message digests of the passwords in the user database

Step 2: User authentication

When a user needs to be authenticated, the user enters the id and password as usual. Now the user's computer computes the message digest of the password and sends user id and

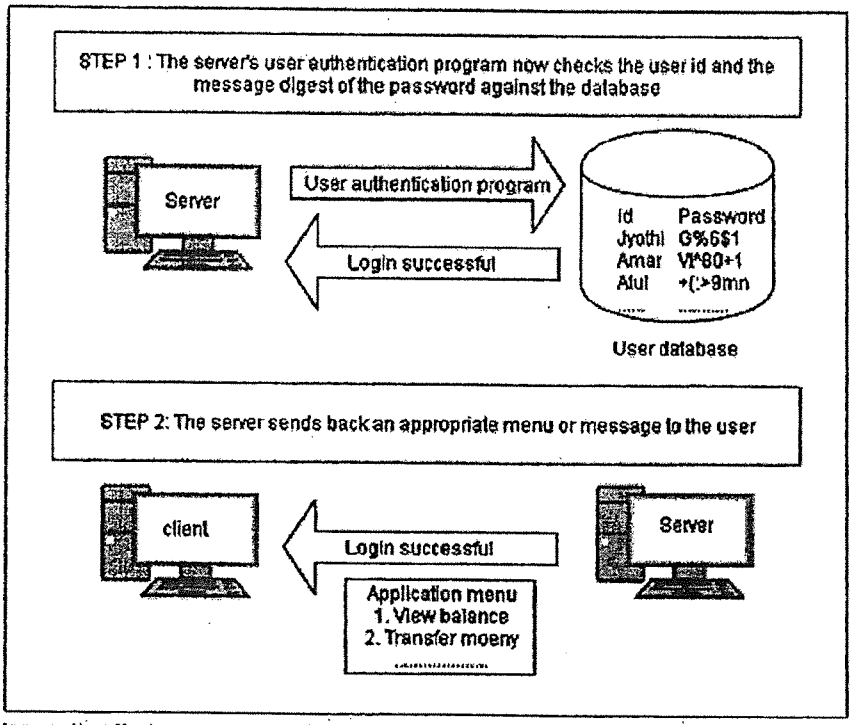
message digest of the password to the server for authentication. This is shown in the following figure:



User authentication involving message digest of the password

Step3: Server-side validation

The user id and message digest of the password travel to the server over communication channel. The server passes these values to the user authentication program which validates the user id and message digest of the password against the database and returns an appropriate response to the back to the server. The server used the result of this operation to return an appropriate message back to the user. This is shown in the following figure:



User authenticator program validates the user id and the message digest of the password

Adding Randomness:

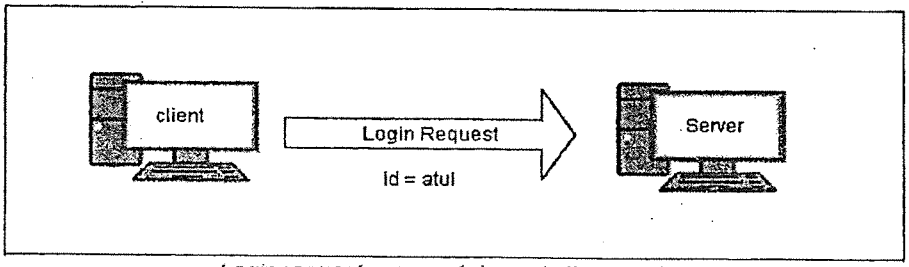
Each time the message digest algorithm produces the same output for the same password. So it creates a problem of ***Replay Attack*** because the attacker simply replays the sequence of actions of a normal user. To overcome from this a random number is added to the user's password. So the message digest algorithm produces different outputs to same password in different login times. This is achieved using following steps:

Step 1: Storing message digests as derived passwords in the user database

We simply store the message digest values of passwords instead of original in the database.

Step 2: User sends a login request

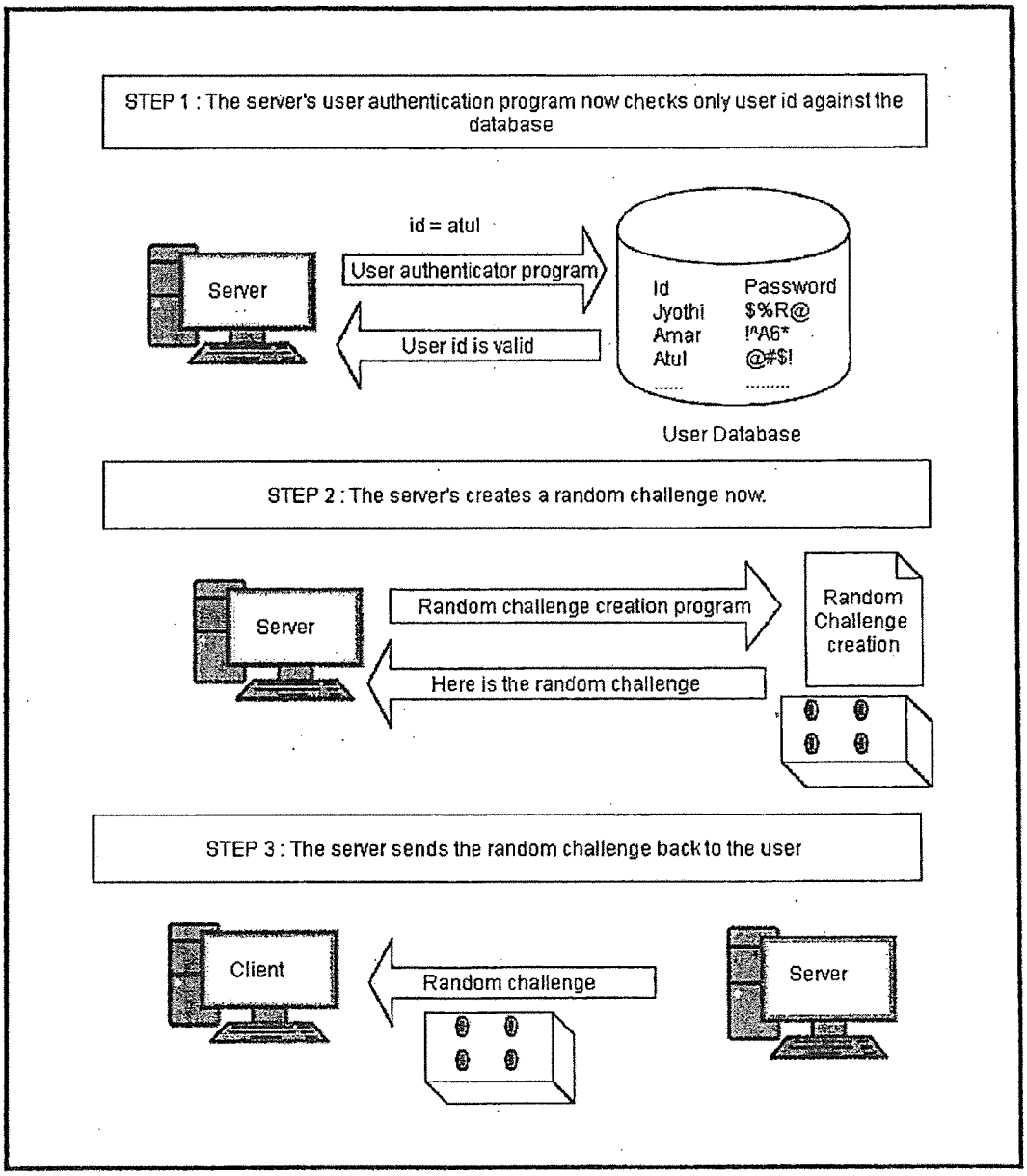
The user sends the login request only with her user id. This is shown in the following figure:



Login request now contains only the user id

Step 3: Server creates a random challenge

When the server receives the user's login request containing the user id alone, it firast checks whether it is valid or not. If it is not valid, an error message is passed to the user. Otherwise the server creates a random challenge (plain text) and sends to user. This is shown in the following figure:

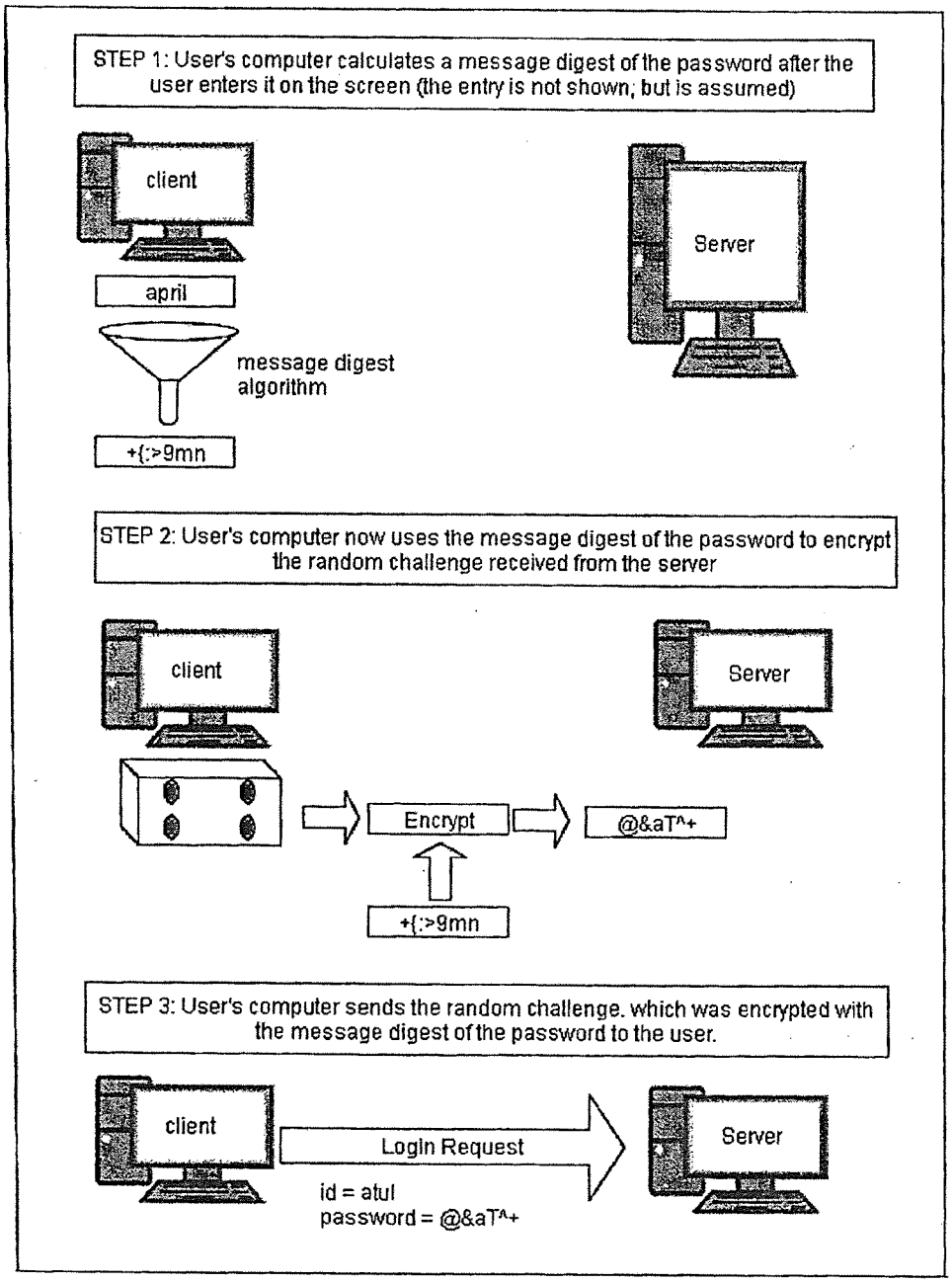


Random challenge creation and transmission to the user

Step 4: User signs the random challenge with message digest of the password

The application displays the password entry screen to the user and the user enters the password. The application executes the message digest algorithm on this password. The

message digest value is used to encrypt the random challenge. This encryption uses symmetric key and is explained in the following figure:



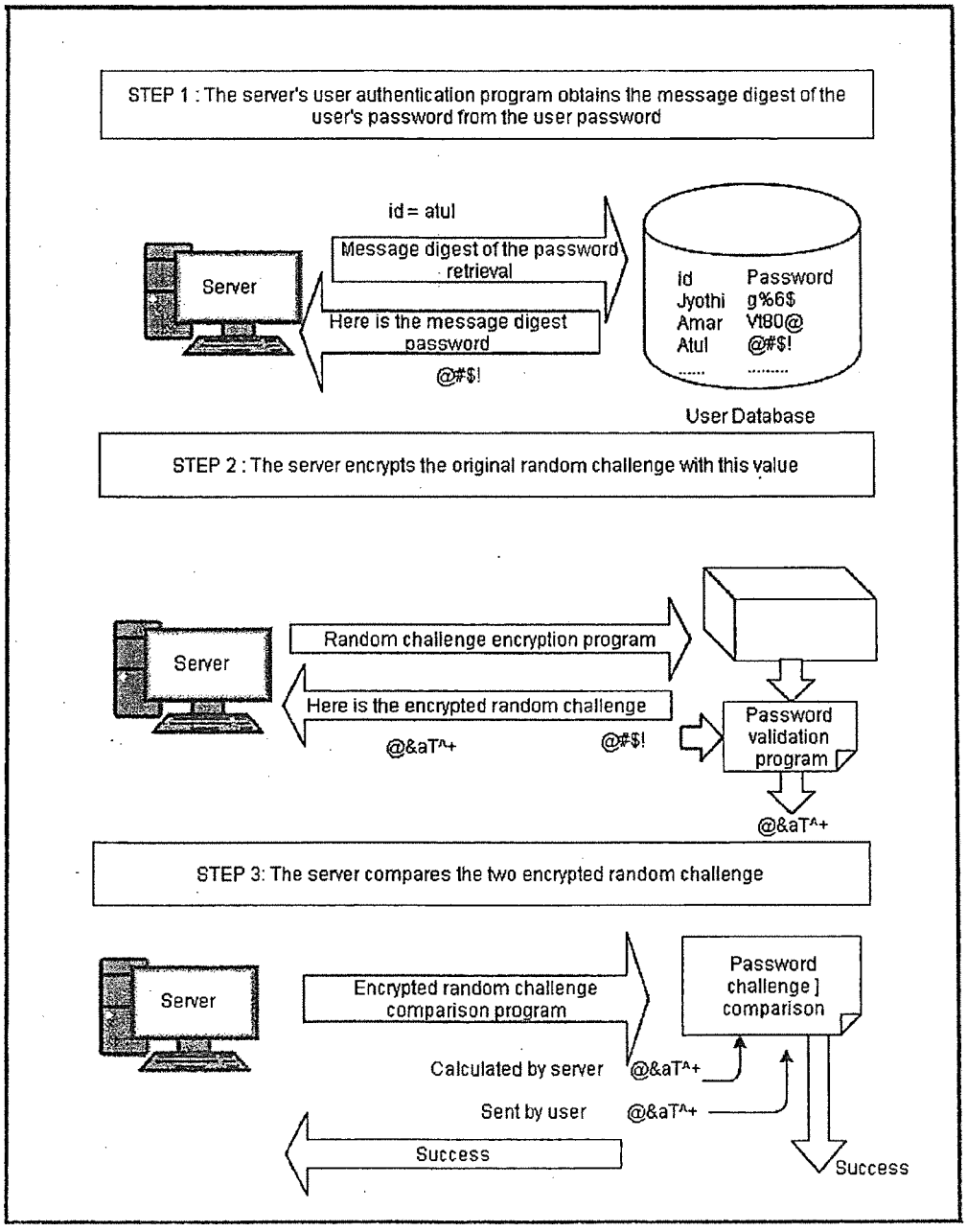
User sends the user id and encrypted random challenge to the server

Step 5: Server verifies the encrypted random challenge received from the user

To verify the encrypted random challenge the server uses one of the following ways:

a. The server can decrypt the received random challenge message from the user. If the decrypted message matches with original random challenge, the server sends success information to the user.

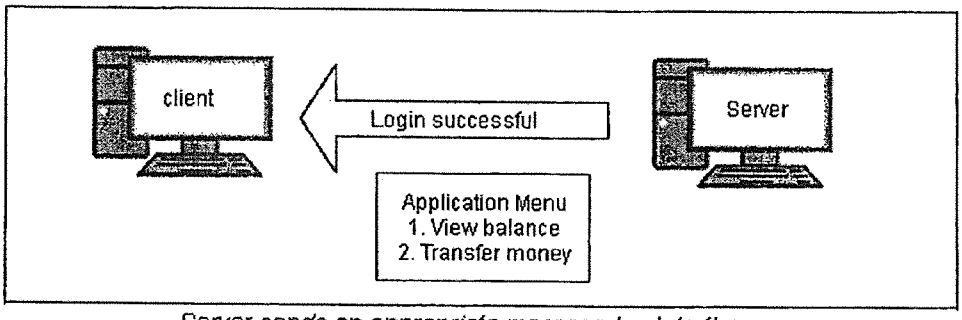
b. The server simply encrypts its own version of the random challenge (sent earlier to the user) with the message digest of the user's password. If this encrypted message matches with user's sent encrypted message, the server sends success information to the user. This approach is shown in the following figure:



Server compares the two encrypted random challenges

Step 6: Server returns an appropriate message back to the user

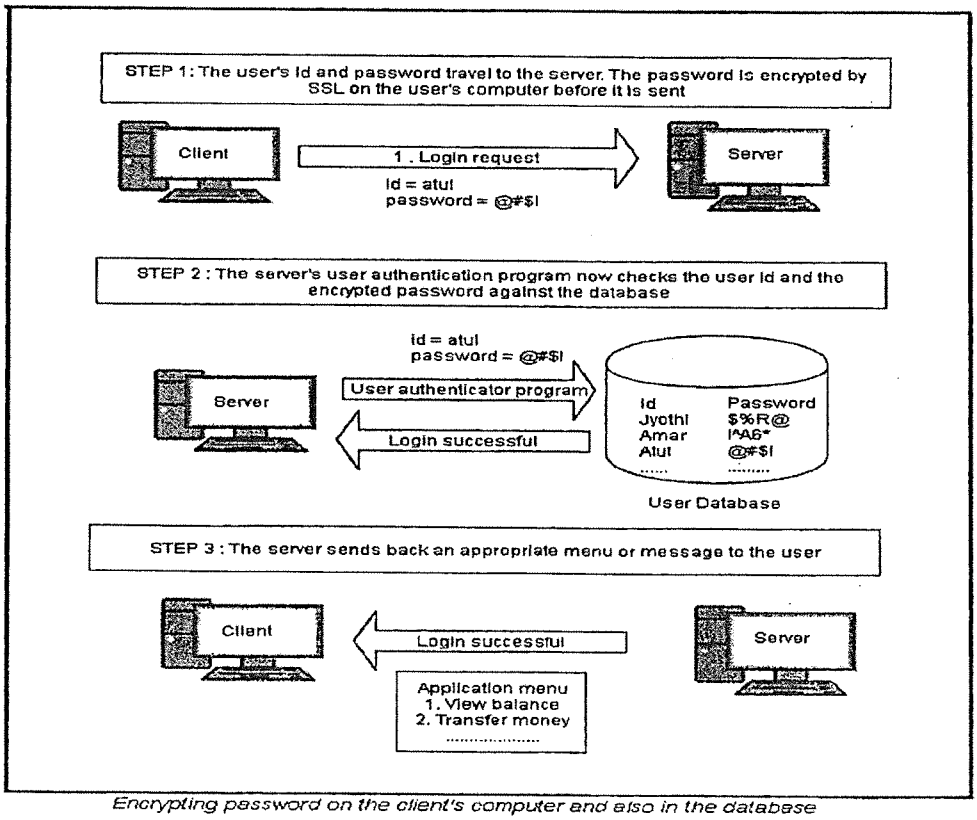
Finally the server sends success or failure information back to the user. This is shown in the following figure:



Server sends an appropriate message back to the user

Password Encryption:

The clear text password is encrypted on the user's computer and then sends it to the server for authentication. In the case of Internet applications, the client is a Web browser, which is not having encryption capability. Consequently, we must resort to technologies such as Secure Socket Layer (SSL). The SSL creates a secure connection between client and server. The SSL would perform the required encryption operations. This is shown in the following figure:



Encrypting password on the client's computer and also in the database

The Problems with Passwords:

The passwords are not truly random:

1. With 52 upper- and lower-case letters, 10 digits and 32 punctuation symbols, there are 948 6 quadrillion possible 8-character passwords
2. Humans like to use dictionary words, human and pet names 1 million common passwords

The problems with passwords are:

1. ***External Disclosure:*** Password becomes known to an unauthorized person by a means outside normal network or system operation includes storing passwords in unprotected files or posting it in an unsecured place.
2. ***Password Guessing:*** Results from very short passwords, not changing passwords often, allowing passwords which are common words or proper names, and using default passwords.
3. ***Live Eavesdropping:*** Results from allowing passwords to transmit in an unprotected manner.
4. ***Verifier Compromise:*** Occurs when the verifier's password file is designed via an internal attack.
5. ***Replay:*** Recording and later replaying of a legitimate authentication exchange.

Types of Attacks:

There are two types of attacks. They are a) On-line Password Attack b) Off-line or Dictionary Password Attack.

1. **On-line password Attack:** One way of guessing passwords is simply to type passwords at the system that is going to verify the password is known as *on-line password attacking*. But the system can make it impossible to guess too many passwords.

   Ex: ATM card eat our card, if we type three incorrect passwords. If the attacker is trying with several different passwords to break the original password, then the system dispatch a message, human to investigate.
2. **Off-line Attack:** An intruder can get a quantity X that is derived from a password in known way. Now the intruder uses an arbitrary amount of compute power to guess passwords and convert them into known way. In this manner the intruder can produce the X. A source of good passwords kept in a little dictionary and the intruder can use to get the X value. Therefore, an off-line password guessing is also known as *Dictionary Attack*.

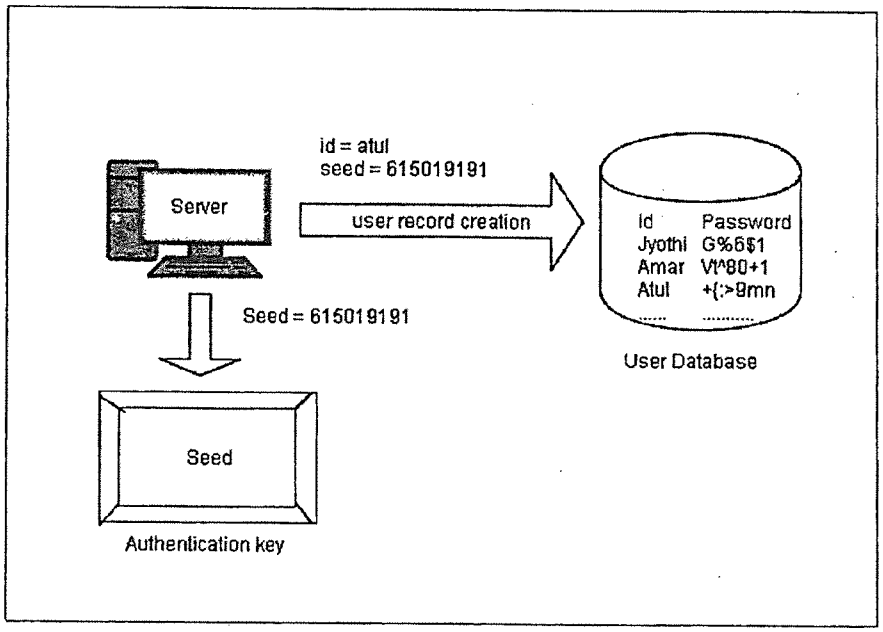
Authentication Tokens:

An authentication token is an extremely useful alternative to a password. An authentication token is a small device that generates a new random value every time it is used. This random value becomes the basis for authentication. The small devices are typically of the size of small key chains, calculators or credit cards. The general features of authentication token are:

1. Processor
2. Liquid Crystal Display (LCD) for displaying outputs
3. Battery
4. (Optionally) a small keypad for entering information
5. (Optionally) a real-time clock

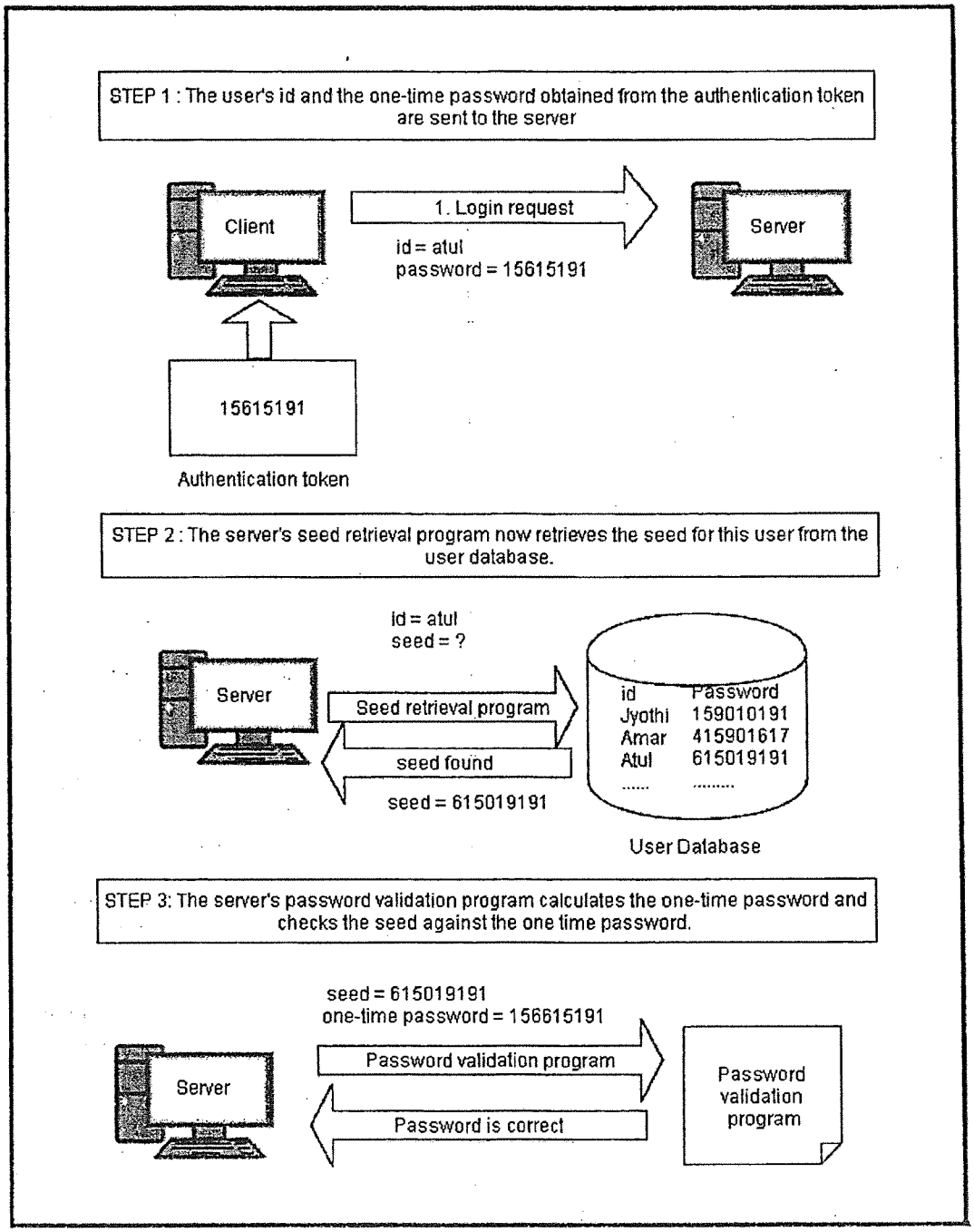
Each authentication token (i.e. each device) is pre-programmed with a unique number called as a **random seed** or just **seed**. The send forms the basis for ensuring the uniqueness of the output produced by the token. This is explained in the following steps:

Step 1: Creation of a token: Whenever an authentication token is created, the corresponding random seed is generated for the token by the **authentication server**. This seed is stored or pre-programmed inside the token, as well as its entry is made against that user's record in the user database. The seed is used automatically by the authentication token. This is shown in the following figure:



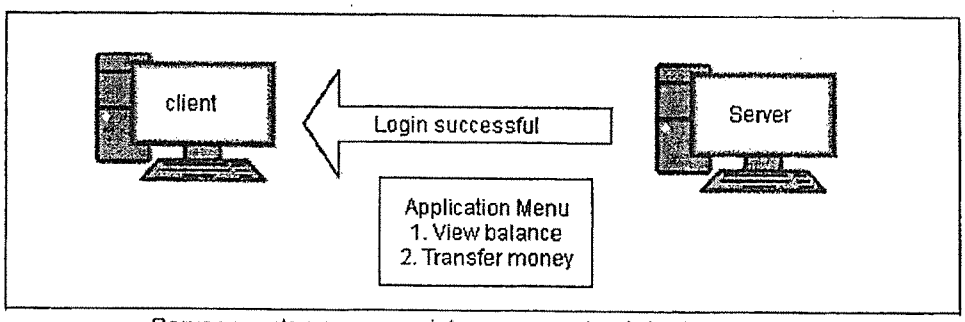
Random seed storage in the database and the authentication token

Step 2: Use of a token: An authentication token automatically generates pseudorandom numbers called as **one-time passwords** or **one-time passcodes**. One-time passwords are generated randomly by an authentication token based on the seed value that they are pre-programmed with. They are one-time because they are generated, used once and discarded forever. This is shown in the following figure:



Server validates the one time password

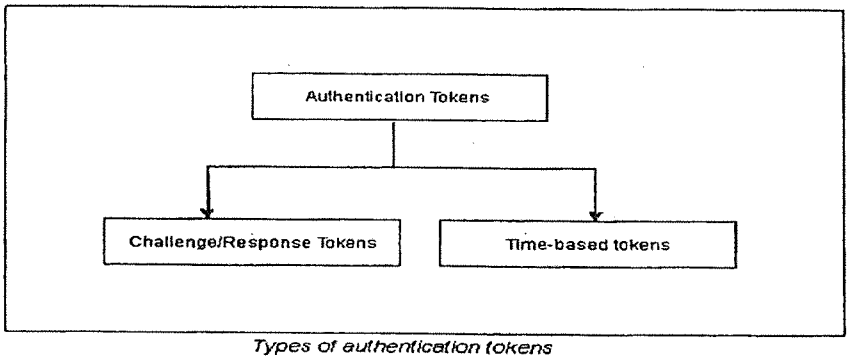
Step 3: Server returns an appropriate message back to the user: Finally, the server sends an appropriate message back to the user depending on whether the previous operations yielded success or failure. This is shown in the following figure:



Server sends an appropriate message back to the user

Authentication Token Types:

There are two main types of authentication tokens. This is shown in the following figure:

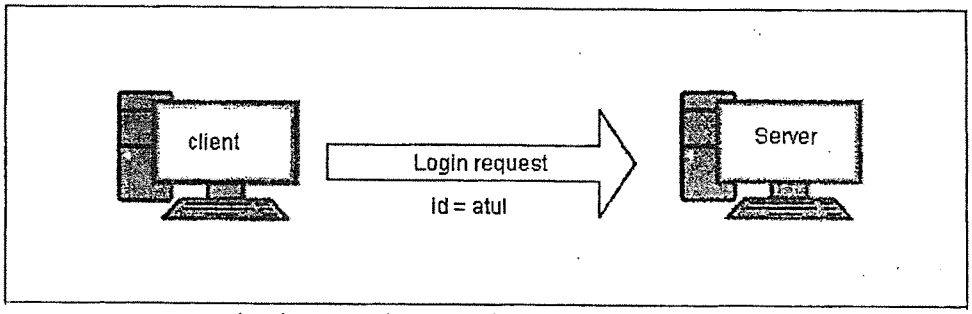


Types of authentication tokens

1. Challenge/Response Tokens:

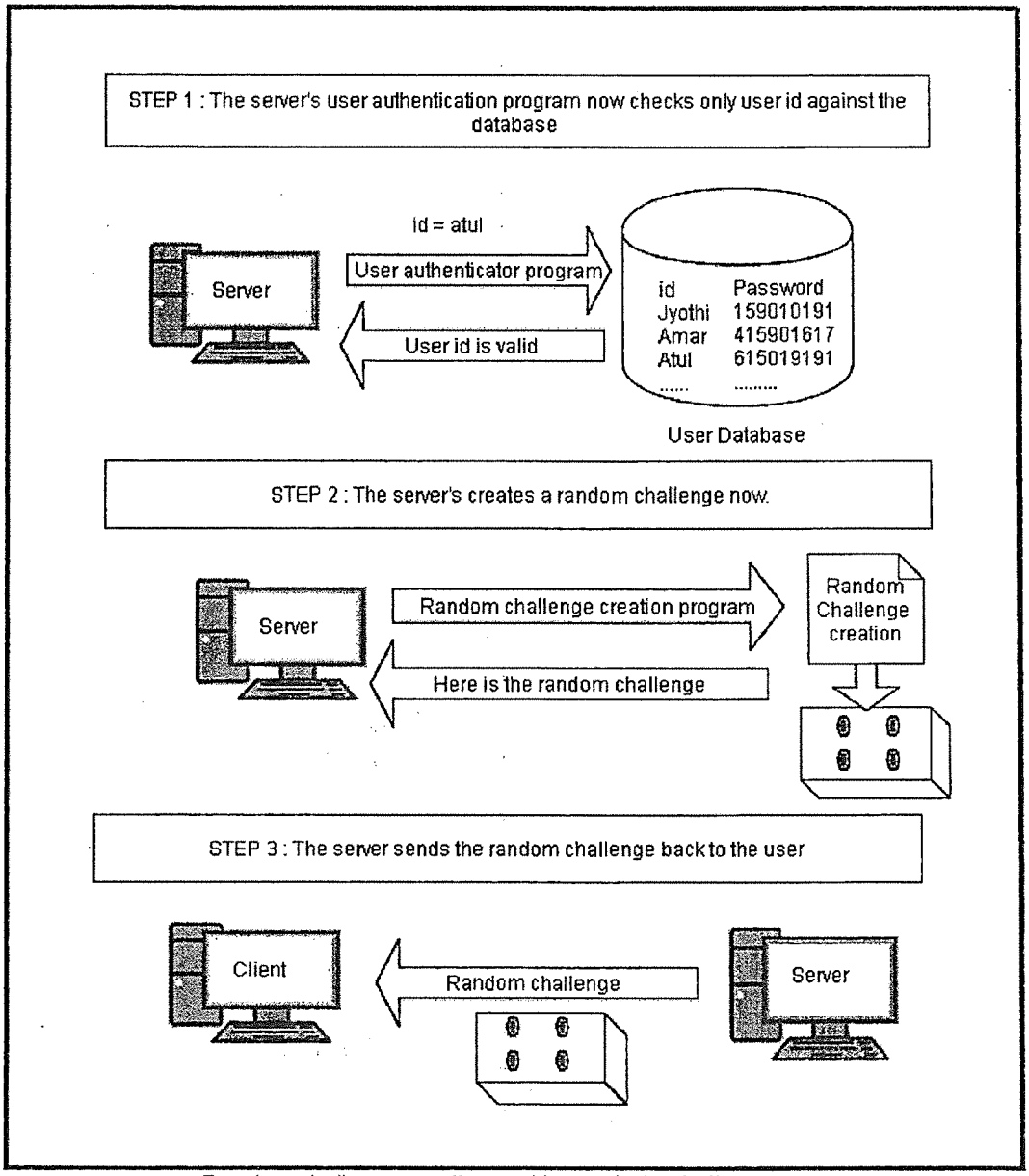
The seed pre-programmed inside an authentication is secret and unique. This fact is basis for the challenge/response tokens. The seed becomes the encryption key in this technique.

Step 1: User sends a login request: The user sends the login request only with user id. This is shown in the following figure:



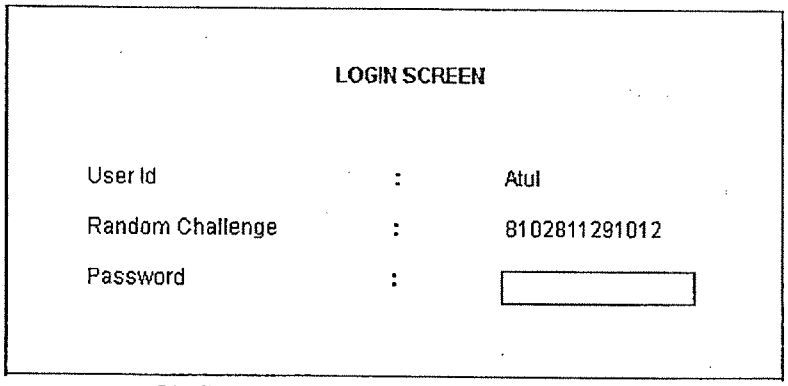
Login request now contains only the user id

Step 2: Server creates a random challenge: It checks user id is validity. If it is invalid, it sends an appropriate message to the user. Otherwise, the server now creates a random challenge (random number) and sends it back to the user. The random challenge can travel as plain text from the server to the user's computer. This is shown in the following figure:



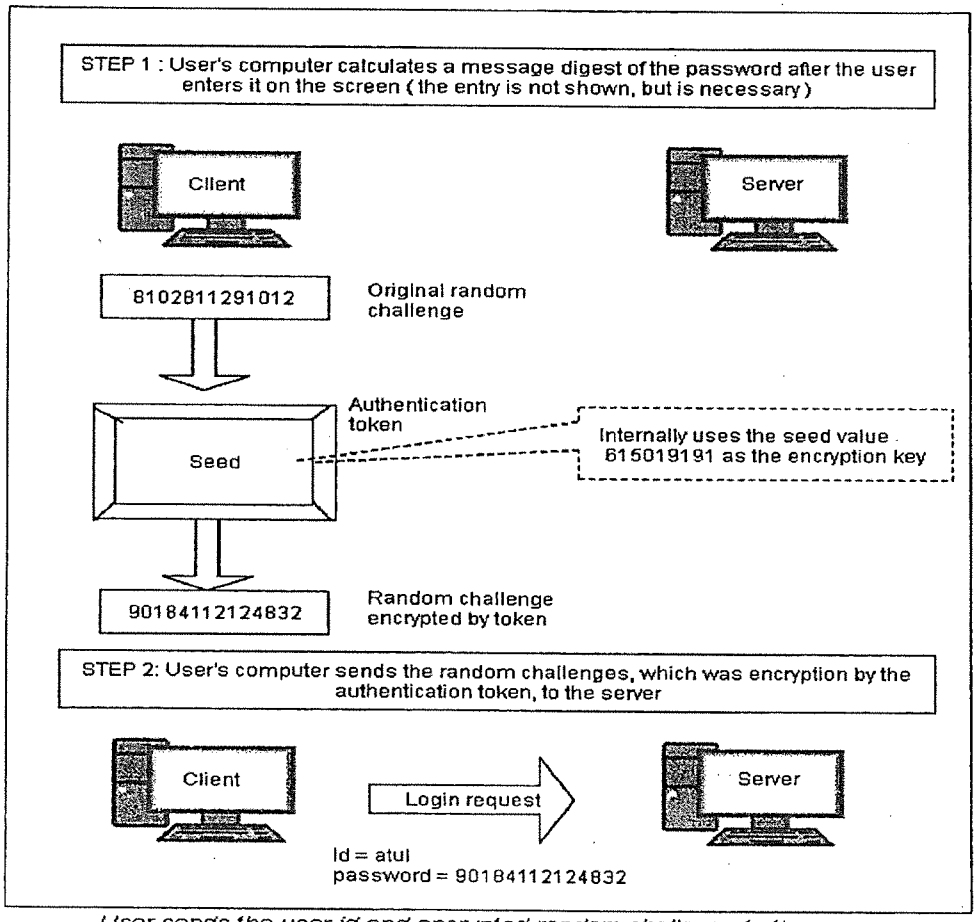
Random challenge creation and transmission to the user

Step 3: User signs the random challenge with the message digest of the password: The user gets the following screen. Let us assume that the random challenge sent by the user was 8102811291012.



Challenge/Response tokens login screen

The user gets the random challenge from the server inside the token. In order to do this, the token has a small keypad. The token accepts the random challenge and encrypts it with the seed value, which is known only to it. The result is the random challenge encrypted with the seed. This result is displayed on the screen of the token. The user reads this value and types it in the password field on the screen. This request is sent to the server as the login request. This is shown in the following figure:

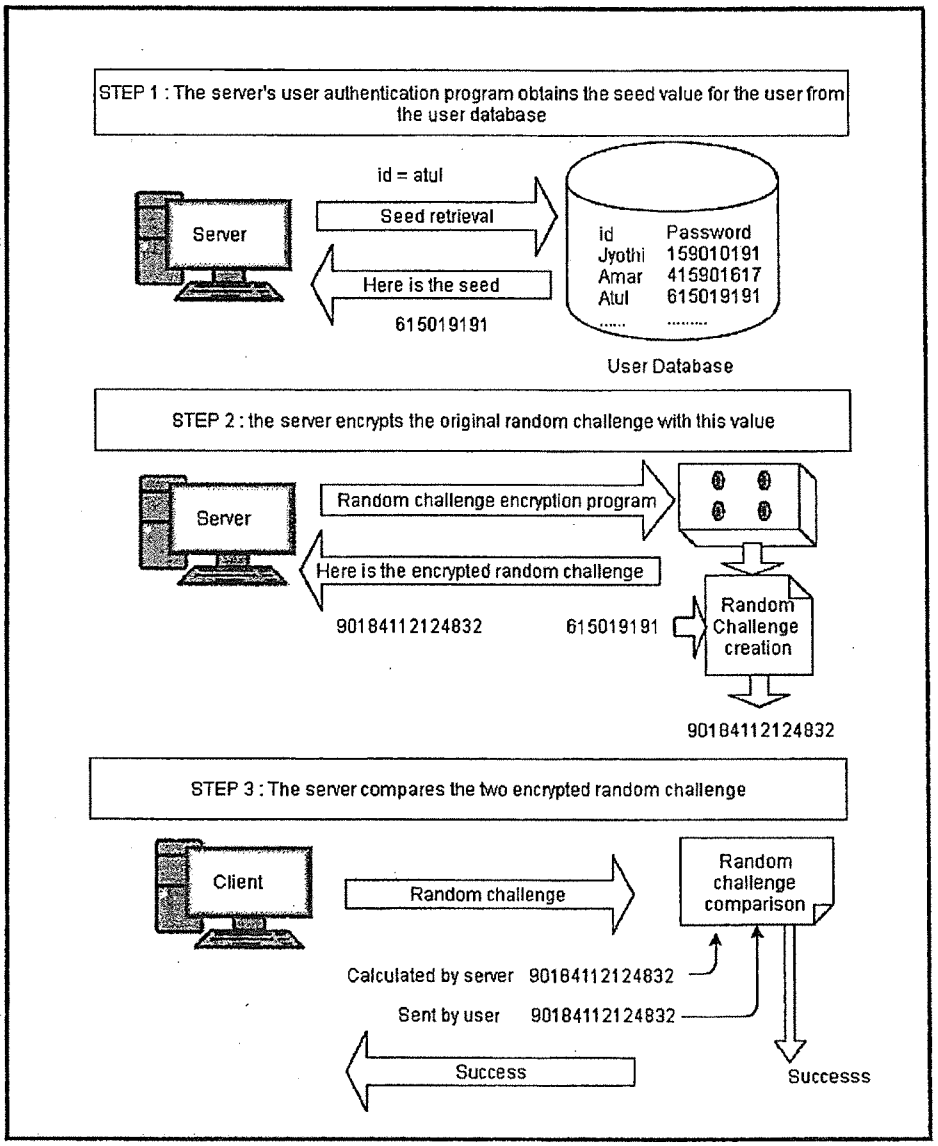


User sends the user id and encrypted random challenge to the server

Step 4: Server verifies the encrypted random challenge received from the user: The received random challenge is verified in two ways:

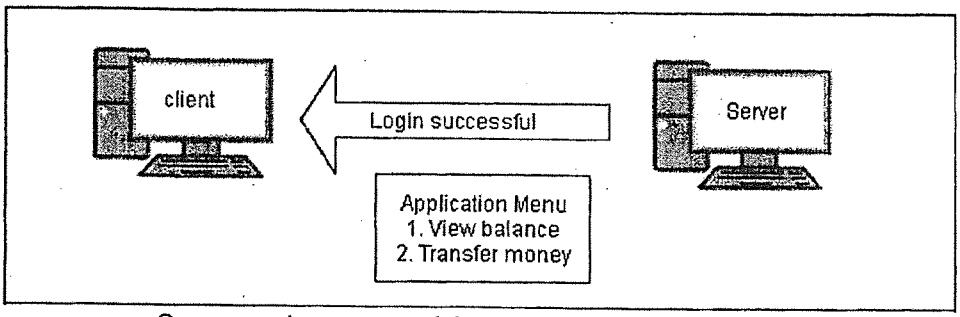
a. The received random challenge is decrypted with seed value for the user. The server verifies the decrypted value which is stored at server.

b. The server can simply encrypt the random number which is stored in it. The encrypted random number is compared with received encrypted random number form the user. This approach is explained in the following figure:



Server compares the two encrypted random challenge

Step 5: Server returns an appropriate message back to the user: Finally the server sends an appropriate message (success/failure) to the user. This is shown in the following figure:

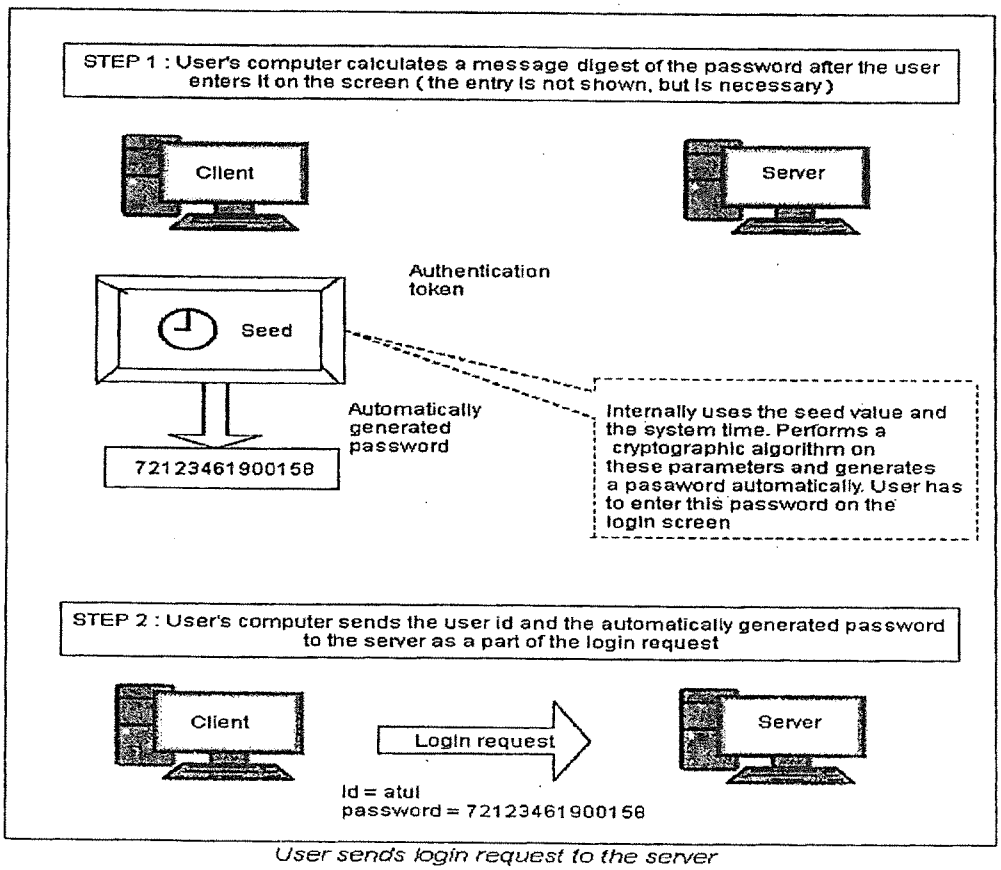


Server sends an appropriate message back to the user

2. Time-based Tokens:

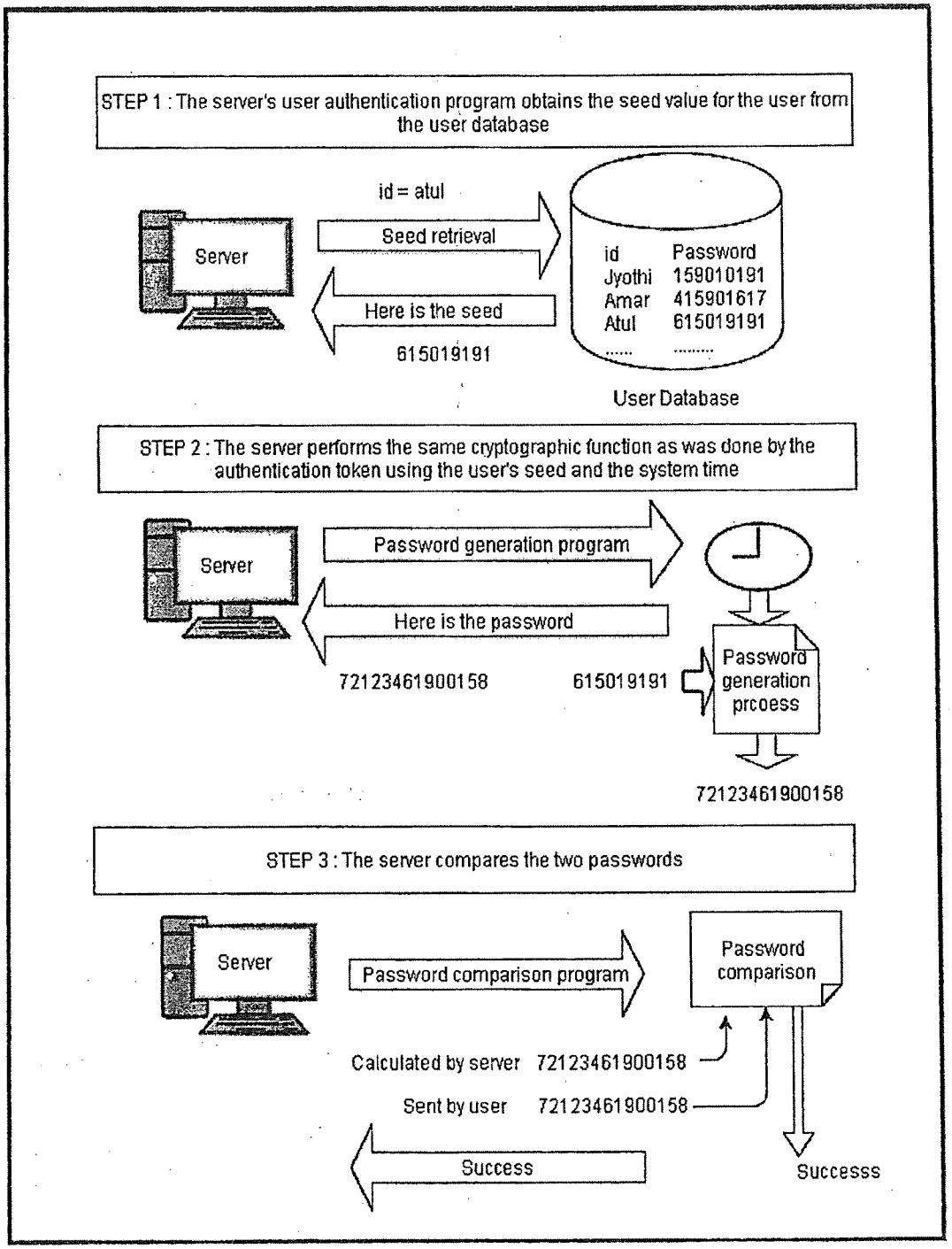
This technique is designed to improve the challenge/response technique by removing drawbacks of it. The process works as follows:

Step 1: Password generation and login request: The tokens automatically generate a password for every 60 seconds and display the latest password on the LCD output for the user to read and use it. To generate the password this technique uses the send and current system time and uses some cryptographic function to produce the password automatically. Whenever a user wishes to logon, the user looks at LCD display and enters user id along with password. This is shown in the following figure:



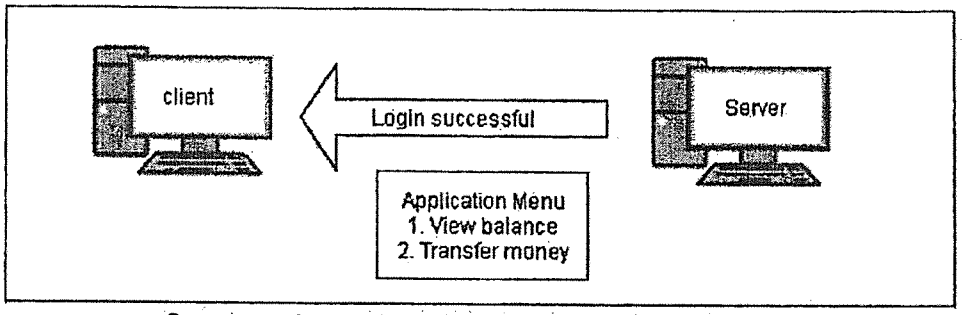
User sends login request to the server

Step 2: Server-side verification: The server verifies the user sent password which is explained in the following figure:



Server verifies the password sent by the user

Step 3: Server returns an appropriate message back to the user: Finally the server sends an appropriate message (success/failure) to the user. This is shown in the following figure:

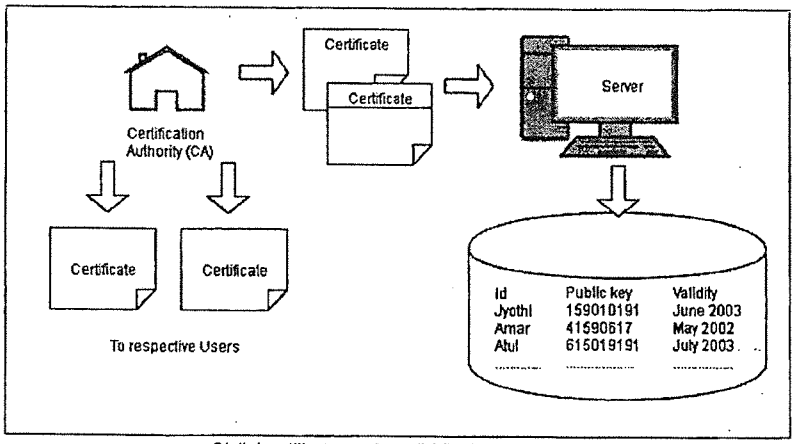


Server sends an appropriate message back to the user

Certificate-Based Authentication:

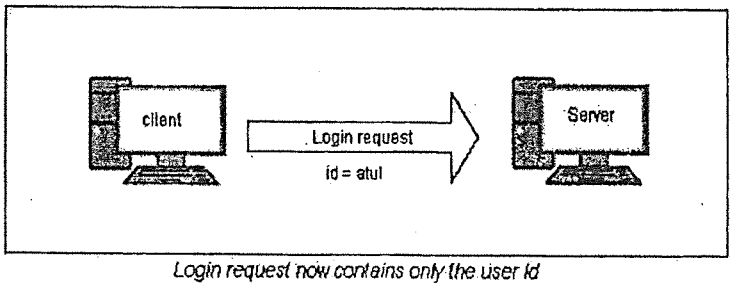
This is a stronger authentication mechanism as compared to a password based authentication because here the user is expected to have certificate and not know password. At the time of login, the user requested to send his/her certificate to the server over the network. The server verifies the user's the validity of the certificate. The following steps explain this mechanism.

Step 1: Creation, storage and distribution of digital certificates: The digital certificates are created and issued to the users and the copy of the certificates is stored in the server in binary format. This is shown in the following figure:



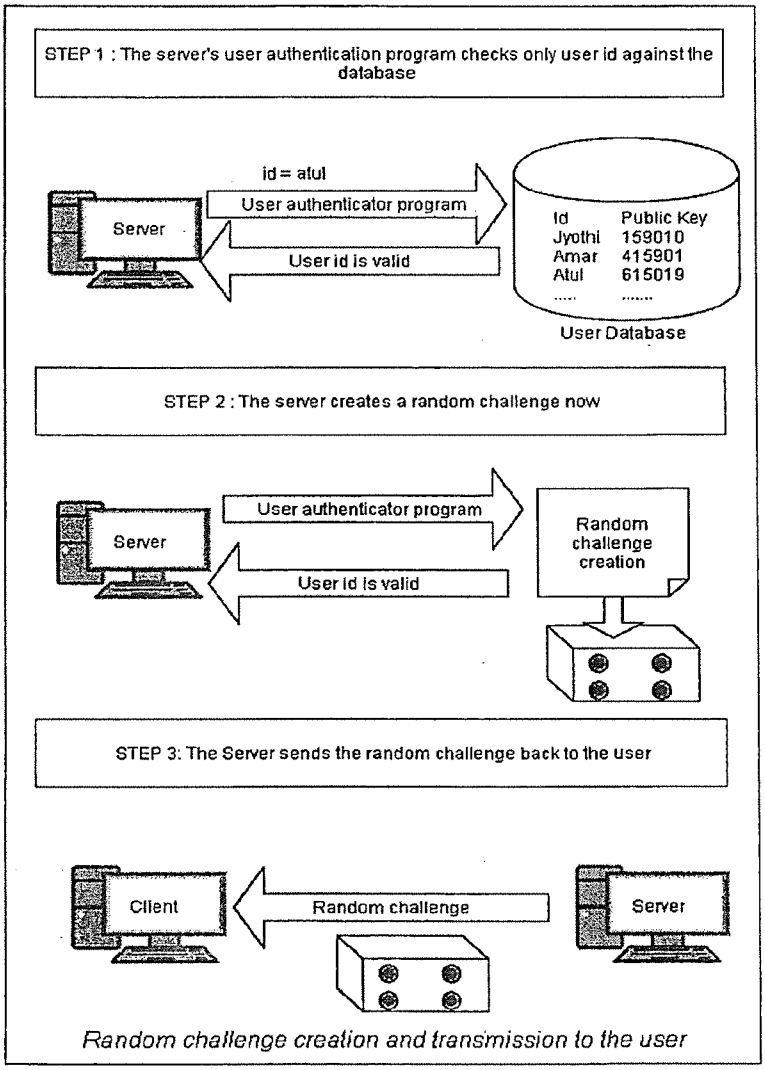
Digital certificate creation, distribution and storage

Step 2: Login request: The user send only user id to the server and is shown in the following figure:



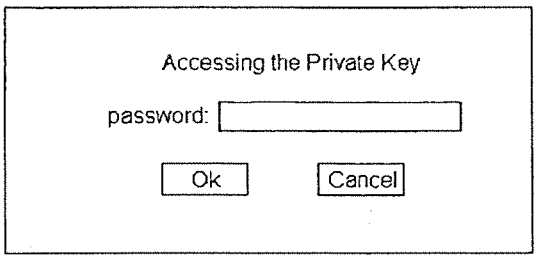
Login request now contains only the user Id

Step 3: Server crates a random challenge: The server validates the user id and if it is correct, it sends a random number as challenge to the user and is explained in the following figure:



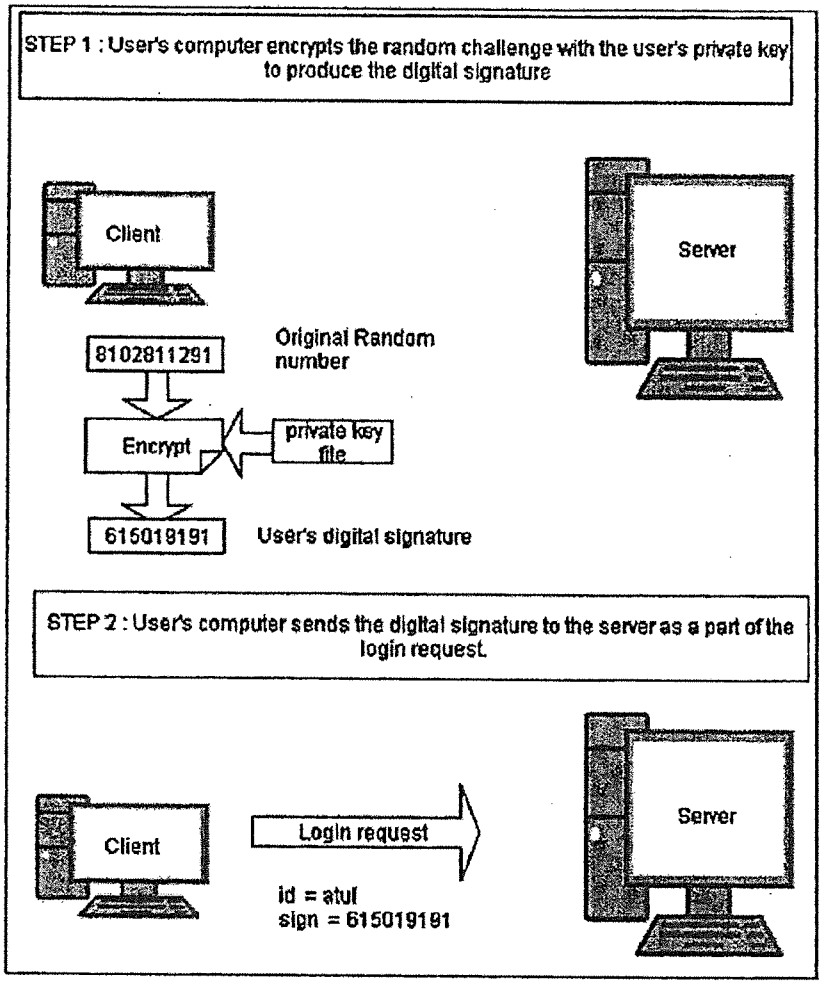
Random challenge creation and transmission to the user

Step 4: User signs the user challenge: The user sings the random number using his/her private key. The user needs to access private key which is stored on disk of his/her computer. This is shown in the following figure:



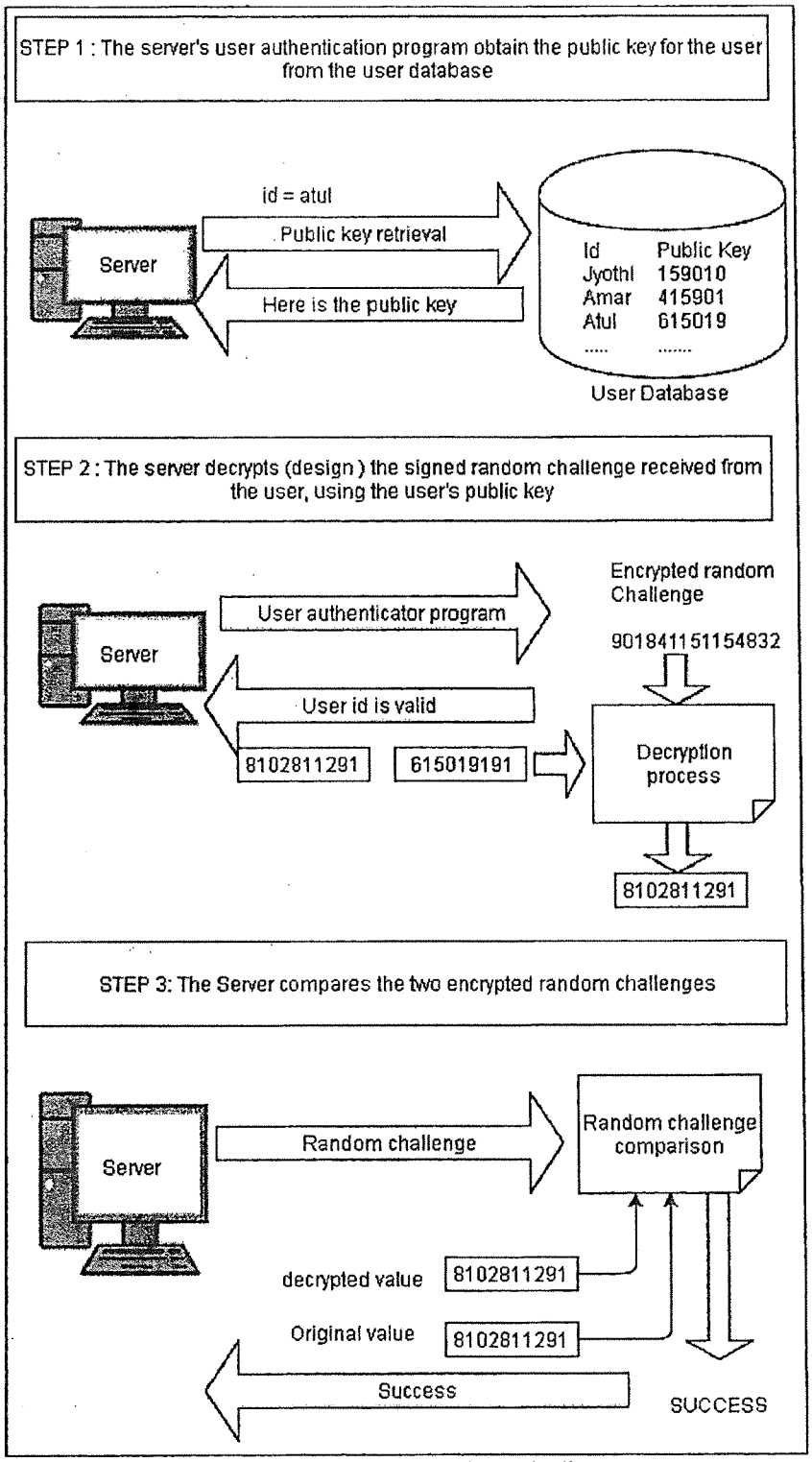
Prompt for password to open private key file

After entering correct password, the user's private key file is opened and the user uses private key to encrypt the random number. This is shown in the following diagram:



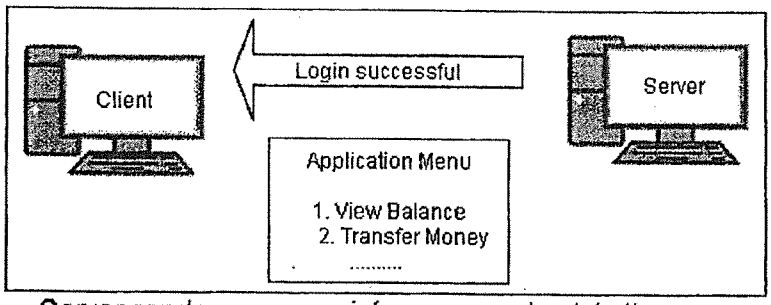
User 's computer signs the random challenge and sends it back to the server

The server uses public key of the user to decrypt the encrypted random number received from the user. This is shown in the following figure:



Server compares the two random challenges

Step 5: Server returns an appropriate message back to the user: Finally the server sends an appropriate message (success/failure) to the user. This is shown in the following figure:



Server sends an appropriate message back to the user

Use of Smart Cards:

The use of smart cards can actually be related to certificate-based authentication because they allow the generation of private-public key pairs within the cards. They also support the storage of digital certificates within the card. The private key always is stored within the card in a secure, tampered free fashion. The public key and the certificate can be exported outside. The smart cards are capable of performing cryptographic functions such as encryption & decryption, message digest creation and signing within the card. The following table explains the problems and their solutions related to smart card technology:

Problem/Issue	*Emerging solution*
Smart card readers are not yet a part of a desktop computer, unlike a hard disk drive or a floppy disk drive	The new versions of computers and mobile devices are expected to come with smart card readers *out of the box*.
Non-availability of smart card reader driver software	Microsoft has made the PC/SC smart card framework an integral part of the Windows 2000 operating system. Most smart card reader manufacturers ship the PC/SC compliant reader drivers, making the process of adding a reader hardware to the computer a plug-and-play operation.
Non availability of smart card aware cryptographic services software	Smart-card aware software such as Microsoft Crypto API (MS-CAPI) comes free with Internet Explorer.
Cost of smart cards and card readers is high	This is reducing now. Smart cards are available for about $5, and the card readers for about $20.

Biometric authentication:

These mechanisms are receiving a lot of public attention. A biometric device is perhaps the ultimate attempt in trying to prove who you are. Biometrics allows a person to be identified and authenticated based on a set of recognizable and verifiable data, which are unique and specific to them.

Biometric authentication is the process of comparing data for the person's characteristics to that person's biometric "template" in order to determine resemblance. The reference model is first store in a database or a secure portable element like a smart card. The

data stored is then compared to the person's biometric data to be authenticated. Here it is the person's identity which is being verified.

In this mode, the question being asked is: "Are you indeed Mr or Mrs X?"

Biometric identification consists of determining the identity of a person. The aim is to capture an item of biometric data from this person, for example by taking a photo of their face, by recording their voice, or by capturing an image of their fingerprint. This data is then compared to the biometric data of several other persons kept in a database.

In this mode, the question being asked is a simple one: "Who are you?"

Types of biometric authentication technologies:

1. **Retina scans** produce an image of the blood vessel pattern in the light-sensitive surface lining the individual's inner eye.
2. **Iris recognition** is used to identify individuals based on unique patterns within the ring-shaped region surrounding the pupil of the eye.
3. **Finger scanning,** the digital version of the ink-and-paper fingerprinting process, and works with details in the pattern of raised areas and branches in a human finger image.
4. **Finger vein ID** is based on the unique vascular pattern in an individual's finger.
5. **Facial recognition** systems work with numeric codes called faceprints, which identify 80 nodal points on a human face.
6. **Voice identification** systems rely on characteristics created by the shape of the speaker's mouth and throat, rather than more variable conditions.

Note: **Hash Functions and SHA-1 are explained in UNIT: II**

2. System Security

Intruders:

One of the two most publicized threats to security is the intruder (the other is viruses), generally referred to as a hacker or cracker. The intruders can classify into three types:

1. **Masquerader:** An individual who is not authorized to use the computer and who makes a way into a system's access controls to exploit a legal user's account
2. **Misfeasor:** A legal user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges
3. **Clandestine user:** An individual who grasps supervisory control of the system and uses this control to avoid auditing and access controls or to hold back audit collection

Intrusion Techniques:

The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. The password file can be protected in one of two ways:

1. **One-way function:** The system stores only the value of a function based on the user's password. When the user presents a password, the system transforms that password and compares it with the stored value. In practice, the system usually performs a one-way transformation (not reversible) in which the password is used to generate a key for the one-way function and in which a fixed-length output is produced.
2. **Access control:** Access to the password file is limited to one or a very few accounts.

Password Guessing:

On the basis of a survey of the literature and interviews with a number of password crackers, reports the following techniques for learning passwords:

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

7. Use a Trojan horse to bypass restrictions on access.
8. Tap the line between a remote user and the host system.

The first six methods are various ways of guessing a password. Guessing attacks are feasible, and indeed highly effective, when a large number of guesses can be attempted automatically and each guess verified, without the guessing process being detectable.

The seventh method of attack i.e., the Trojan horse can be particularly difficult to counter. The eighth attack listed i.e., line tapping, is a matter of physical security. It can be countered with link encryption techniques.

Password Management:

Password Protection:

The front line of defense against intruders is the password system. The password serves to authenticate the ID of the individual logging on to the system. The ID provides security in the following ways:

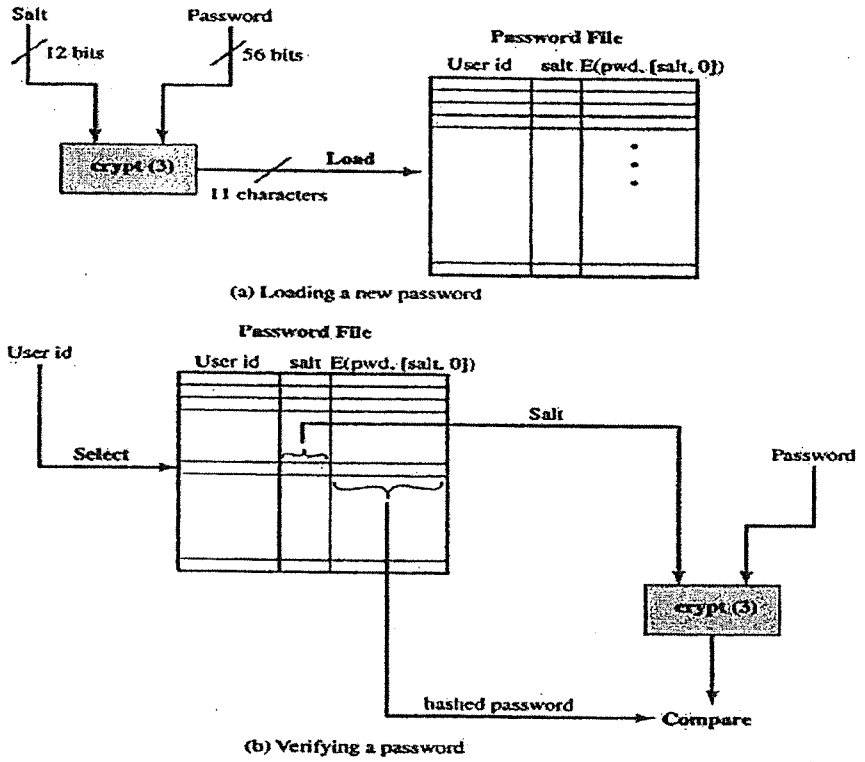
1. The ID determines whether the user is authorized to gain access to a system. In some systems, only those who already have an ID filed on the system are allowed to gain access.
2. The ID determines the privileges accorded to the user. A few users may have supervisory or "superuser" status that enables them to read files and perform functions that are especially protected by the operating system. Some systems have guest or anonymous accounts, and users of these accounts have more limited privileges than others.
3. The ID is used in what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

UNIX Password Scheme:

To understand the nature of the threat to password-based systems, let us consider a scheme that is widely used on UNIX, in which passwords are never stored in the clear.

1. Each user selects a password of up to eight printable characters in length.
2. This is converted into a 56-bit value (using 7-bit ASCII) that serves as the key input to an encryption routine.
3. The encryption routine, known as crypt(3), is based on DES.
4. The DES algorithm is modified using a 12-bit "salt" value. Typically, this value is related to the time at which the password is assigned to the user.

5. The modified DES algorithm is exercised with a data input consisting of a 64-bit block of zeros.
6. The output of the algorithm then serves as input for a second encryption.
7. This process is repeated for a total of 25 encryptions. The resulting 64-bit output is then translated into an 11-character sequence.
8. The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID.
9. This method has been shown to be secure against a variety of cryptanalytic attacks. This is explained in the following diagram:



The salt serves three purposes:

1. It prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned at different times. Hence, the "extended" passwords of the two users will differ.
2. It effectively increases the length of the password without requiring the user to remember two additional characters. Hence, the number of possible passwords is increased by a factor of 4096, increasing the difficulty of guessing a password.

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

There are two threats to the UNIX password scheme. They are

1. A user can gain access on a machine using a guest account or by some other means and then run a password guessing program, called a password cracker, on that machine.
2. The cracker simply has to test the password file against lists of likely passwords. Because many people use guessable passwords, such a strategy should succeed on virtually all systems.

Password Selection Strategies:

There are four techniques for password selection. They are

1. **Using Computer Generated Passwords:** UNIX operating system provides some predefined passwords which is mixture of alphabets and numerals. Even if the password is pronounceable, it is difficult to remember by the user.
2. **Educating users in Password selection:** It is required to educate the users to design a strong password.
3. **Reactive Password Checking:** A password that time to time runs its own password cracker to find guessable passwords is Reactive Password Checking. System cancels any passwords that are entered by users is guessed and the same thing notified to the user.
4. **Proactive Password Checking:** The proactive password checker verifies is that password feasible or not i.e., whether the password is a guessable password or not.

Intrusion Detection:

The intrusion detection is motivated by a number of considerations, including the following:

1. If an intrusion is detected quickly enough, the intruder can be identified and turned out from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to preempt the intruder, the earlier that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved.
2. An effective intrusion detection system can serve as a prevention, so acting to prevent intrusions.

3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

Approaches to Intrusion Detection:

The following approaches are used for Intrusion Detection:

1. **Statistical anomaly detection:** Involves the collection of data relating to the behavior of legal users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legal user behavior.
    a. Threshold detection: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
    b. Profile based: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.
2. **Rule-based detection:** Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.
    a. **Anomaly detection:** Rules are developed to detect deviation from previous usage patterns.
    b. **Penetration identification:** An expert system approach that searches for suspicious behavior.

Audit Records:

A fundamental tool for intrusion detection is the audit record. Basically, two plans are used:

1. **Native audit records:** Virtually all multiuser operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form.
2. **Detection-specific audit records:** A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine.

Statistical Anomaly Detection:

Statistical anomaly detection techniques fall into two broad categories: threshold detection and profile-based systems.

1. Threshold detection involves counting the number of occurrences of a specific event type over an interval of time. If the count exceeds what is considered a reasonable number that one might expect to occur, then intrusion is assumed. Threshold analysis, by itself, is a simple and unsuccessful detector of even moderately sophisticated attacks.
2. Profile-based anomaly detection focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations. A profile may consist of a set of parameters, so that variation on just a single parameter may not be sufficient in itself to signal an alert.

Rule-Based Intrusion Detection:

Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious. There are two types of approaches for rule-based intrusion detection. They are

1. **rule-based anomaly detection**
   - 1.1. analyze historical audit records to identify usage patterns & auto-generate rules for them
   - 1.2. then observe current behavior & match against rules to see if conforms
   - 1.3. like statistical anomaly detection does not require prior knowledge of security flaws
2. **rule-based penetration identification**
   - 2.1. uses expert systems technology
   - 2.2. with rules identifying known penetration, weakness patterns, or suspicious behavior
   - 2.3. rules usually machine & O/S specific
   - 2.4. rules are generated by experts who interview & codify knowledge of security admins
   - 2.5. quality depends on how well this is done
   - 2.6. compare audit records or states against rules

The Base-Rate Fallacy:

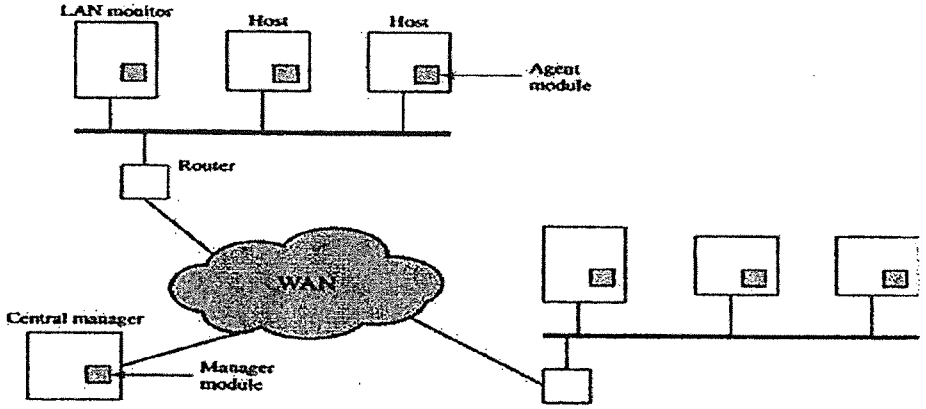
1. practically an intrusion detection system needs to detect a large percentage of intrusions with few false alarms
   1.1. if too few intrusions detected -> false security
   1.2. if too many false alarms -> ignore / waste time
2. this is very hard to do
3. existing systems seem not to have a good record

Distributed Intrusion Detection:

Generally, intrusion detection systems focused on single-system. But in general we have networked systems (LAN). So a more effective defense has these working together to detect intrusions. The main issues are

a. dealing with varying audit record formats
b. integrity & confidentiality of networked data
c. centralized or decentralized architecture

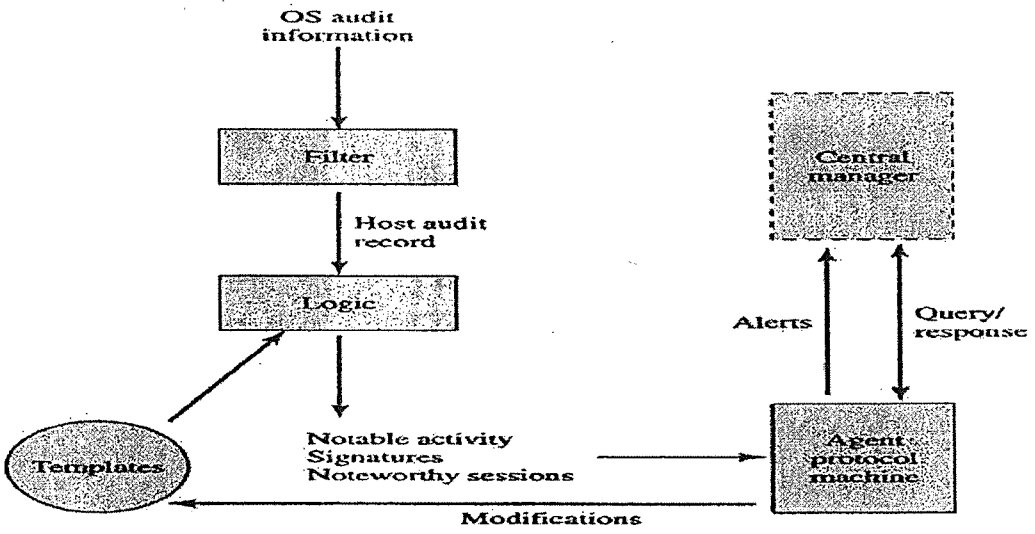
The following distributed intrusion detection system diagram is developed at the University of California at Davis [HEBE92, SNAP91].



The above diagram shows the overall architecture, which consists of three main components:

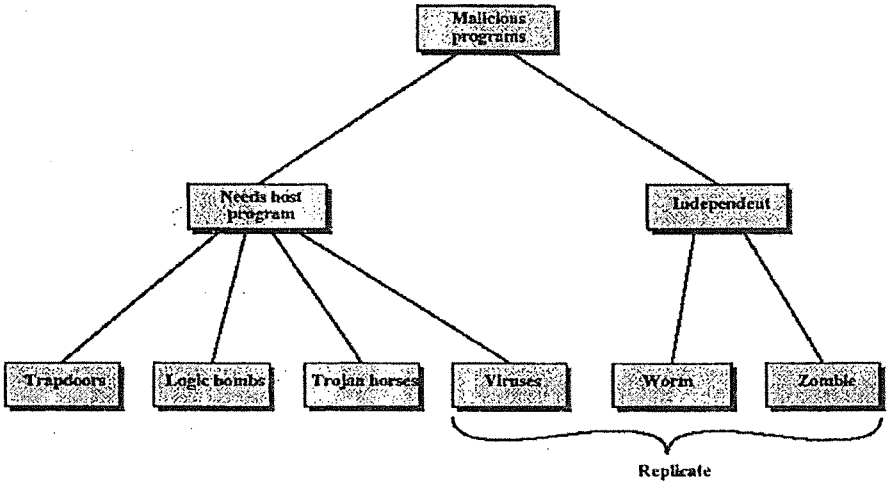
1. **Host agent module:** An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on security-related events on the host and transmit these to the central manager.
2. **LAN monitor agent module:** Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.
3. **Central manager module:** Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion.

The scheme is designed to be independent of any operating system or system auditing implementation. The agent captures each audit record produced by the native audit collection system. The following diagram shows the general approach of agent architecture.



Malicious Software:

Malicious software is software that is intentionally included or inserted in a system for a harmful purpose. The following diagram provides classification of software threats or malicious programs.



1. **Trapdoor:** An undocumented entry point intentionally written into a program, often for debugging purposes, which can be exploited as a security flaw.
2. **Trojan Horse:** Instructions hidden inside an otherwise useful program that do undesirable things.

3. **Logic Bomb:** Malicious instructions that trigger on some event in the future, such as a particular time occurring.
4. **Virus:** A set of instructions that, when executed, inserts copies of itself into other programs.
5. **Worm:** A program that replicates itself by installing copies of itself on other machines across a network. Similar to a bacterium, but it replicates over a network whereas a bacterium tends to stay confined to one machine. Examples are:
   a. Electronic mail facility: A worm mails a copy of itself to other systems.
   b. Remote execution capability: A worm executes a copy of itself on another system.
   c. Remote login capability: A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other.
6. **Bacterium:** A free-standing program that replicates itself, causing harm by consuming resources.
7. **Zombie:** Malicious instructions installed on a system that can be remotely triggered to carry out some attack with less traceability because the attack comes from another victim. Often the attacker installs large number of zombies to generate large bursts of network traffic.

The Nature of Viruses:

A virus is a piece of software that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs. Once a virus is executing, it can perform any function, such as erasing files and programs. During its lifetime, a typical virus goes through the following four phases:

1. **Dormant phase:** The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.
2. **Propagation phase:** The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.
3. **Triggering phase:** The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.

4. **Execution phase:** The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

Virus Structure:

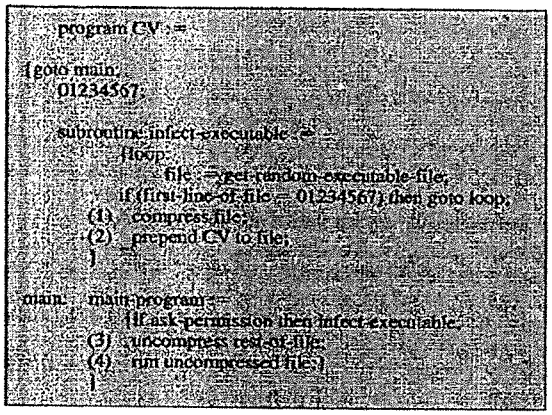
A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.

A very general description of virus structure is shown as the following. 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.

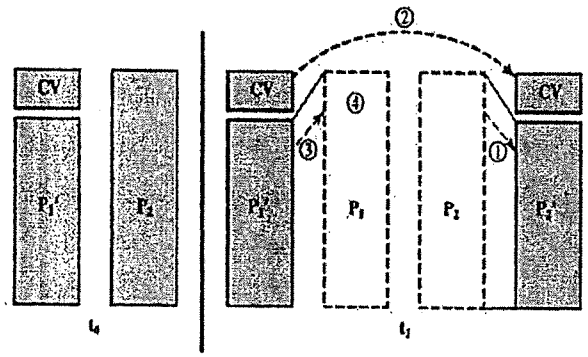
```
program V :=
{goto main;
 1234567;

 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:
}
```

When the following program is invoked, control passes to its virus, which performs the following steps:

a. For each uninfected file P2 that is found, the virus first compresses that file to produce P'2, which is shorter than the original program by the size of the virus.
b. A copy of the virus is prepended to the compressed program.
c. The compressed version of the original infected program, P'1, is uncompressed.
d. The uncompressed original program is executed.



Logic for a Compression Virus



A Compression Virus

In the above example, the virus does nothing other than propagate. As in the previous example, the virus may include a logic bomb.

Initial Infection:

Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes. Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place.

Types of Viruses:

The viruses are classified into six types. They are

1. **Parasitic virus:** The traditional and still most common form of virus. A parasitic virus attaches itself to executable files and replicates, when the infected program is executed, by finding other executable files to infect.
2. **Memory-resident virus:** Lodges in main memory as part of a resident system program. From that point on, the virus infects every program that executes.

3. **Boot sector virus:** Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.
4. **Stealth virus:** A form of virus explicitly designed to hide itself from detection by antivirus software. (a virus that uses compression so that the infected program is exactly the same length as an uninfected version)
5. **Polymorphic virus:** A virus that mutates with every infection, making detection by the "signature" of the virus impossible. (A **polymorphic virus** creates copies during replication that are functionally equivalent but have distinctly different bit patterns)
6. **Metamorphic virus:** As with a polymorphic virus, a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration, increasing the difficulty of detection. Metamorphic viruses may change their behavior as well as their appearance.

Macro Viruses:

In the mid-1990s, macro viruses became the most common type of virus. Macro viruses are particularly threatening for a number of reasons:

1. A macro virus is platform independent. Virtually all of the macro viruses infect Microsoft Word documents. Any hardware platform and operating system that supports Word can be infected.
2. Macro viruses infect documents, not executable portions of code. Most of the information introduced onto a computer system is in the form of a document rather than a program.
3. Macro viruses are easily spread. A very common method is by electronic mail.

Macro viruses take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro. Basically, a macro is an executable program embedded in a word processing document or other type of file.

Antivirus Approaches:

The ideal solution to the threat of viruses is prevention: Though prevention is better than cure, it is impossible to achieve. The next best approach is to be able to do the following:

1. **Detection:** Once the infection has occurred, determine that it has occurred and locate the virus.
2. **Identification:** Once detection has been achieved, identify the specific virus that has infected a program.

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

[STEP93] identifies four generations of antivirus software. They are

1. **First generation: simple scanners:** A first-generation scanner requires a virus signature to identify a virus. The virus may contain "wildcards" but has essentially the same structure and bit pattern in all copies. Such signature-specific scanners are limited to the detection of known viruses. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length.
2. **Second generation: heuristic scanners:** This scanner uses heuristic rules to search for likely virus infection. One class of such scanners looks for fragments of code that are often associated with viruses. Another second-generation approach is integrity checking. A checksum can be appended to each program. If a virus infects the program without changing the checksum, then an integrity check will catch the change.
3. **Third generation: activity traps:** This generation programs are memory-resident programs that identify a virus by its actions rather than its structure in an infected program.
4. **Fourth generation: full-featured protection:** This generation products are packages consisting of a variety of antivirus techniques used in conjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.

Advanced Antivirus Techniques:

The most important advanced antivirus techniques are

1. **Generic Decryption:** Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses, while maintaining fast scanning speeds. This contains the following elements:
   a. **CPU emulator:** A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor. The emulator includes software versions of all registers and other processor hardware, so that the underlying processor is unaffected by programs interpreted on the emulator.

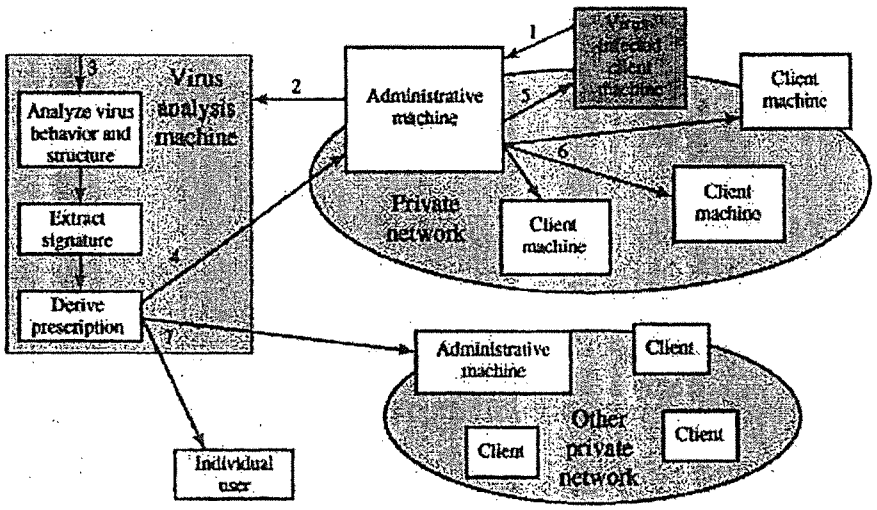
b. **Virus signature scanner:** A module that scans the target code looking for known virus signatures.

c. **Emulation control module:** Controls the execution of the target code.

2. **Digital Immune System:** The digital immune system is a comprehensive approach to virus protection developed by IBM. The motivation for this development has been the rising threat of Internet-based virus propagation. Two major trends in Internet technology have had an increasing impact on the rate of virus propagation in recent years:

   a. **Integrated mail systems:** Systems such as Lotus Notes and Microsoft Outlook make it very simple to send anything to anyone and to work with objects that are received.

   b. **Mobile-program systems:** Capabilities such as Java and ActiveX allow programs to move on their own from one system to another.



Digital Immune System

The above Figure illustrates the typical steps in digital immune system operation:

i. A monitoring program on each PC uses a variety of heuristics based on system behavior, suspicious changes to programs, or family signature to infer that a virus may be present. The monitoring program forwards a copy of any program thought to be infected to an administrative machine within the organization.

ii. The administrative machine encrypts the sample and sends it to a central virus analysis machine.

iii. This machine creates an environment in which the infected program can be safely run for analysis. Techniques used for this purpose include emulation, or the

creation of a protected environment within which the suspect program can be executed and monitored. The virus analysis machine then produces a prescription for identifying and removing the virus.

iv. The resulting prescription is sent back to the administrative machine.

v. The administrative machine forwards the prescription to the infected client.

vi. The prescription is also forwarded to other clients in the organization.

vii. Subscribers around the world receive regular antivirus updates that protect them from the new virus.

Trusted Systems:

Sometimes we need to protect our data on the basis of levels, i.e., entire data is accessible for MD (Managing Director), some other data is accessible for Managers and some other data is accessible by Accounts Department. This is generally happens in the defense where we have to classify the data into confidential, secret, top secret data.

Trusted system provides such type of Multilevel Security. The Multilevel Security is defined in the following way. In the Multilevel Security, we have subjects, objects, access rights and two important principles. 1) No Read up 2) No Write Down

Subject: An entity that wants to access the data or object the subject may be any user or any processes.

Object: Anything which is accessed by subject (For example, a database or file). An object is nothing but what we want to provide security.

Access Rights: The way in which an object is accessed by the subject is known as access right. For example, Read, Write and Execute. The access rights are specified in the form of an access matrix. It is as shown in the following:

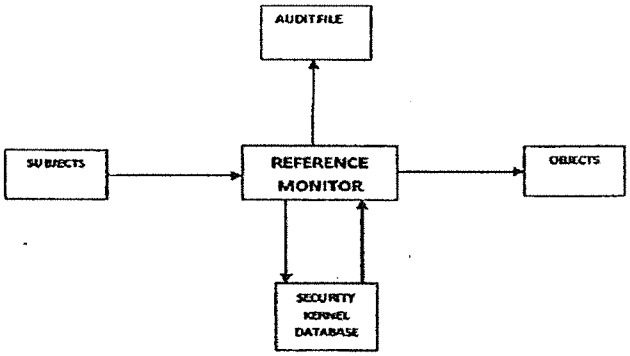
Object / Subject	File1	File2	DB1	DB2	Seg. A	Seg. B
User 1	R	W	RW	RWE		
User 2	W		R	E		RWE
Process 1	R	W	E	E	R	W
Process 2	RW	WE	ER	R	W	E

There are the two principles that are used in the Multilevel Security.

1. **No Read up:** A subject can only read an object of same security level or lower security level and it ca not read an object at higher level. This property is also known as Simple Security level. This property is also known as Simple Security Policy.

2. **No Write down:** A subject can only write into an object at equal or higher level security and it cannot write into an object at lower security level. This property is also known as *- property.

If these two security policies fully described along with the access rights then Multilevel Security will be provided. The trusted system based on reference monitor concept will provide Multilevel Security. This is shown in the following diagram:



Here, the Reference Monitor controls the access of objects by the Subjects according to the security parameters that are defined in the security kernel database. Here, all the access, no read up & no write down principles are programmed in the security kernel database. The following are the properties of Reference Monitor.

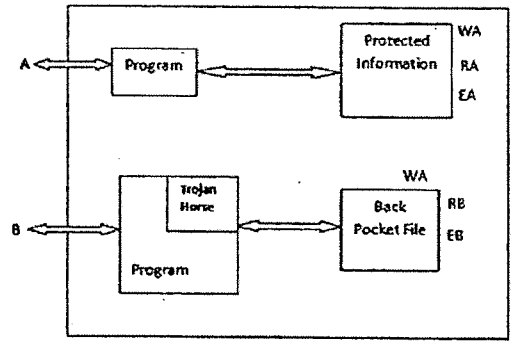
1. ***Complete Mediator*:** The security rules are enforced (applied) to each & every request.
2. ***Isolation*:** The reference monitor and security kernel database must be protected from unauthorized user. These are under control of administrator.
3. ***Verifiability*:** Each and every decision that are taken by the reference monitor is not changed for a request until kernel database modified.

As the reference monitor is capable of providing verifiability, we call it as a ***trusted system***. The final element in the reference monitor concept Audit File.

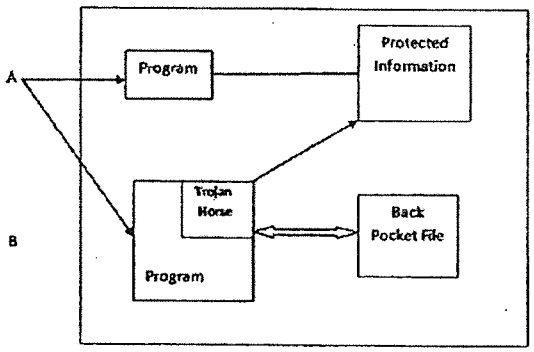
Audit File: The reference monitor keeps all the transactions that are granted or that are stored in the audit file. It also keeps the information about illegal operations performed by the users in the audit file. All the security policies that are having some drawbacks are also written in the audit file. Based on the audit file the administrators may update the kernel database.

Trojan Horse and Defense: Trojan horse attack begins when an illegal user wants to access the information from a legal users login before him. After a legal user 'A' gains access to system, 'B' comes to him and asks to run a file or program in the floppy. The program internally contains another file called Trojan horse. When the program is executed by the

legal user 'A', internally Trojan horse is also executed. Its job is to create a file called Back Pocket file. On this file, 'Write' right is given to the user who logs in i.e., 'A'. The Trojan Horse program copies all the protected information from the user A's login into back pocket file. After completion of copying the program ends, user 'B' comes with his floppy. Now the floppy contains the back pocket file in which he has the required information. This attack is explained in the following diagram:



First Stage

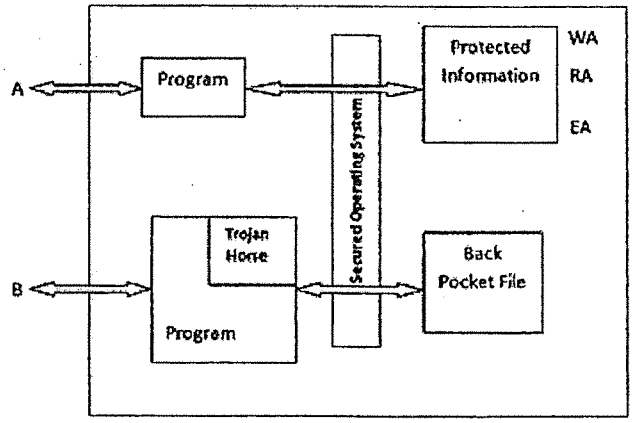


Second Stage

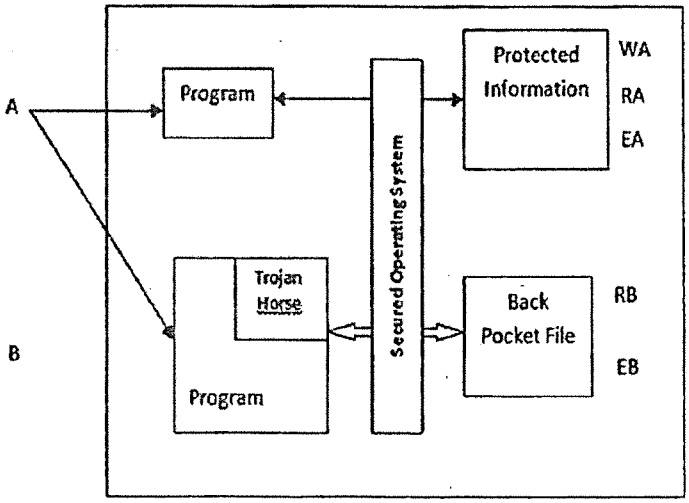
Let us consider the use of secured Operating System in this scheme. Here, we have to define security levels for the subjects and objects. Here, we use two important security levels: They are a) Public b) Sensitive.

Sensitive is higher than Public. Whenever user 'A" logs in his program, his data (protected) are given Sensitive security level. All other programs that he will execute, which are not owned by him are given Public security level. So, user B's program gets public security level. Whenever user 'A' executes B's program, it executes, the Trojan Horse program is also executed. Its job is read information, from the user A which is assigned Sensitive Security level. We have the principle "No Read up", in the security Operating System. So, the Trojan Horse program which is having public security level is not able to

read user A's protected data which is assigned sensitive security level. As a result, Back Pack file is created and nothing written into it. This is shown in the following diagram:



First Stage



Second Stage

****

UNIT - IV

1. Internet Security Protocols: Basic concepts-SSL-SHTTP-TSP-SET-SSL versus SET- 3D secure protocol-Electronic money-Email security-WAP security-security in GSM.

2. Network Security: Brief Introduction to TCP/IP - Firewalls -IP security-Virtual Private Networks.

1. Internet Security Protocols

Q. Define Web Page and Website. Explain types of web pages.

Web page is a document available on World Wide Web. Web Pages are stored on web server and can be viewed using a web browser. A web page can contain huge information including text, graphics, audio, video and hyperlinks. These hyperlinks are the link to other web pages.

Collection of linked web pages on a web server is known as **website.** There is unique **Uniform Resource Locator (URL)** is associated with each web page. Web pages are classified into three types. They are

1. Static Web Pages: Static web pages are also known as flat or stationary web page. They are loaded on the client's browser as exactly they are stored on the web server. Such web pages contain only static information. User can only read the information but can't do any modification or interact with the information. Static web pages are created using only HTML (Hyper Text Markup Language). Static web pages are only used when the information is no more required to be modified. This is explained in the following figures:

2. Dynamic Web Pages: Dynamic web page shows different information at different point of time. It is possible to change a portion of a web page without loading the entire web page. It has been made possible using **Ajax (Asynchronous JavaScript and XML)** technology. This is shown in the following figure. Dynamic web pages are classified into two types. They are:

a. Server-Side Dynamic Web Page: It is created by using server-side scripting. There are server-side scripting parameters that determine how to assemble a new web page which also includes setting up of more client-side processing.

b. Client-Side Dynamic Web Page: It is processed using client side scripting such as JavaScript and then passed in to **Document Object Model (DOM).**

3. Active Web Pages: An **active** web page is a page where the *browser* performs the logic instead of the server. When a client sends an HTTP request for an active web page, the web server sends back an HTTP response that contains an HTML page as usual. In addition, the HTML page also contains a small program that executes on the client computer inside the web browser. This is shown in the following figure:

Protocol, Internet Protocol and Packet:

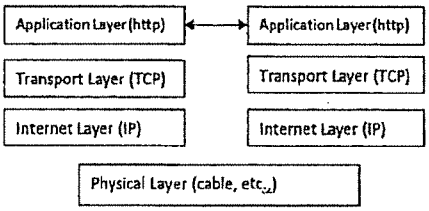
In information technology, a **protocol** is the special set of rules that end points in a telecommunication connection use when they communicate. Protocols specify interactions between the communicating entities.

The **Internet Protocol (IP)** is the method or protocol by which data is sent from one computer to another on the Internet. Each computer (known as a host) on the Internet has at least one IP address that uniquely identifies it from all other computers on the Internet. A packet is the unit of data that is routed between an origin and a destination on the Internet or any other packet-switched network. When any file (e-mail message, HTML file, Graphics Interchange Format file, Uniform Resource Locator request, and so forth) is sent from one place to another on the Internet.

TCP/IP:

The Transmission Control Protocol/Internet Protocol (TCP/IP), is a suite of communication protocols used to interconnect network devices on the internet. TCP/IP can also be used as a communications protocol in a private network (an intranet or an extranet).

The whole Internet runs on these two protocols IP and TCP. These two protocols provide all of the requirements to transmit and receive data across complex WANs (Wide Area Network). Networks are made of layers and each layer providing a specific functionality. The following figure describes layers of a network protocol:

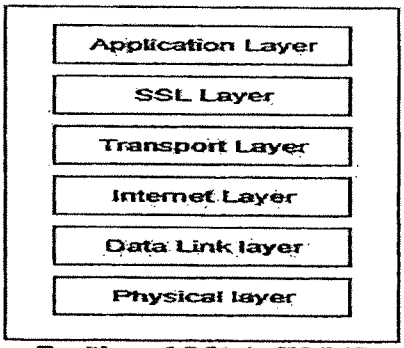


1. The Physical Layer is made from the actual hardware (cables, network interface cards etc.,) and the drivers which are required to run that hardware.
2. The Application Layer represents the application which we are running. In our case the application is the WEB and the application layer is HTTP (*H*yper *T*ext *T*ransfer *P*rotocol).

3. The Transport Layer establishes a reliable communication stream between a pair of systems across a network by putting sequence numbers in packets, holding packets at the destination until they can be delivered in order and retransmitting lost packets.
4. The Internet Layer or **IP layer** is a group of internetworking methods, protocols, and specifications in the Internet protocol suite that are used to transport datagrams (packets) from the originating host across network boundaries.
5. Internet-layer protocols use IP-based packets.

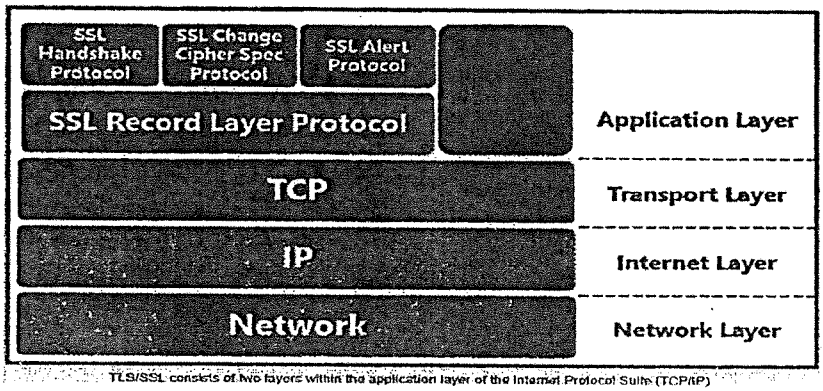
Secure Socket Layer (SSL):

The SSL protocol is an Internet Protocol for secure exchange of information between a web browser and a web server. It provides two basic security services: authentication and confidentiality. Logically, it provides a secure pipe between the web browser and the web server. This is developed by Netscape Corporation developed in 1994 and SSL become the world's most popular web security mechanism. All the major web browsers support SSL. SSL comes in three versions: 2, 3 and 3.1. The most popular version is 3 and is released in 1995. SSL is located in between application and transport layers. This is shown in the following figure:



Position of SSL in TCP/IP

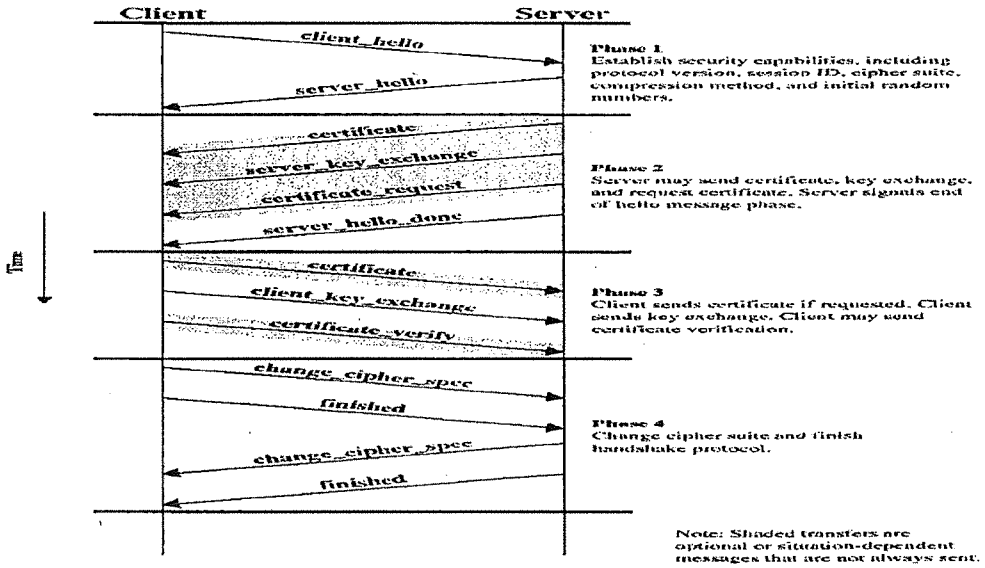
SSL is designed to make use of TCP to provide reliable service. It is not a single protocol. It consists two layers of protocol as shown in the following figure:



TLS/SSL consists of two layers within the application layer of the Internet Protocol Suite (TCP/IP)

SSL record protocol provides basic security services to the higher level protocols. In the higher level, we have HTTP protocol. It provides transfer service for web client server interaction. There are 3 higher level SSL protocols. They are

1. SSL Alert Protocol
2. SSL Change Cipher Sequence Protocol
3. SSL Handshaking protocol. These protocols are used in *management of SSL.*



There are 2 important SSL concepts:

a. SSL Connection b. SSL Session

A connection is a transport that provides transport service. In SSL, the connections are end to end and are associated with a session. A session is created by handshaking protocol. It contains a set of cryptographic security parameters which are shared by the connection.

Between any pair of parties (Systems), there are may be more than one connection and each connection may have its own session. Totally between each pair of parties, there may be simultaneous sessions, but generally we use only one connection between parties & that connection is fully secured. A ***session*** is defined by the following parameters:

a) Session Identifier b) Peer Certificate c) Compression Method d) Cipher Specification e) Master Secret Value

The ***connection*** is defined the following parameters:

1. Client & Server
2. Server Write MAC Secret

3. Client Write MAC Secret
4. Server Write Key – the conditional key for data encryption at server, same key is used in decryption at client.
5. Client Write Key – the connectional key used for encryption at client site, same key used for decryption at sever site.
6. Initialization vectors – When CBC mode is used, an IV is used.
7. Sequence Number – each parity maintains a separate sequence number for sending messages and received messages.

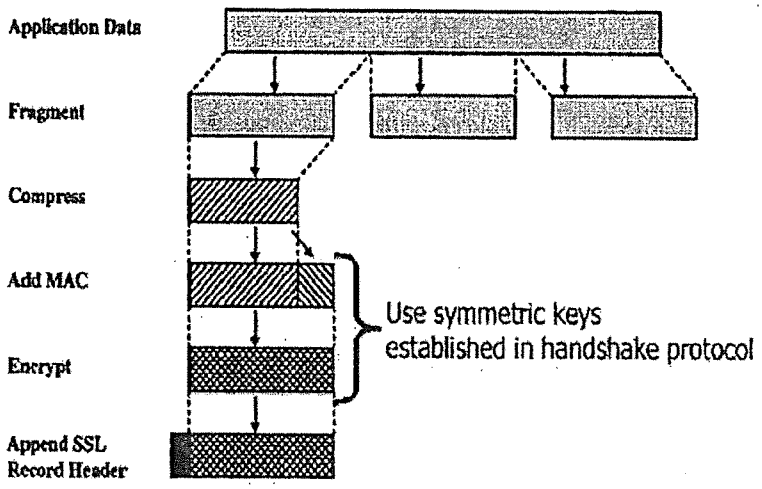
The SSL record protocol provides 2 services for the SSL connections:

1) Confidentiality 2) Message Integrity

The *handshaking protocol* defines a secret key shared between client & server. Later this key is used in encryption of SSL payloads. It provides confidentiality.

The handshaking protocol defines a secret key shared between client & server. Later this key is used to form MAC which provides integrity. The following is the overall operation provided by SSL record protocol.

This protocol takes a message to be transmitted, fragments it into several blocks, optionally comprises each block, apply a MAC, append MAC at the end of each block (block may be compressed or not), encrypt each block. Finally append SSL record header to each block & transmit to destination which is shown in the following figure.



At the receiving end SSL record protocol performs remove header part, decrypt, verifies MAC values, and optionally decompresses to get a fragment. Such fragments are reassembled to get the message & it is delivered to the end user.

In the SSL record protocol, MAC is calculated for the compressed fragment or uncompressed fragment. To calculate this, we require a key shared between two parties. The MAC value is calculated using the following formulae:

hash (MAC_write_secret || pad – 2 || hash (MAC_write_secret || pad -1 || sequence no. || SSL compressed type || SSL compressed length || SSL compressed fragment))

Where hash calculates hash value using MD5 or SHA-1 & write_MAC_secret (or MAC_write) is shared key.

pad -1 – if MD5 algorithm is used 001101110 bits is repeated48 times. If SHA-1 algorithm is used, those are repeated 40 times.

pad-2 – the bits 01011100 repeated 48-times for MD5 algorithm & 40-times for SHA-1 algorithm.

Sequence no. – is unique no. and is used for this message.

SSL compressed type – type of compression technique is used.

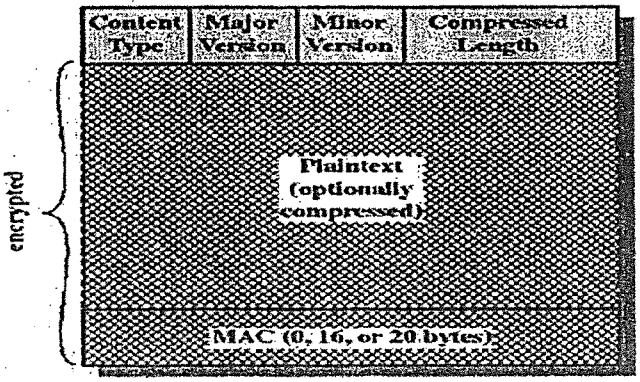
SSL compressed length – is length of the compressed fragment.

SSL compressed fragment – is compressed fragment.

The authentication formulae used here is not discussed so far. This is a variation to HMAC. In HMAC we use XOR operation for the padding bits. Here we use append. Finally, we prepared a header consisting of following fields:

1. content type (8-bits)
2. Major version (8-bits) – it indicates the major version of the SSL protocol used & its value is 3.
3. Minor version – it indicates minor version of SSL protocol used & its value is 0.
4. Compressed length – indicates length of the compressed fragment after padding MAC.

The following figure shows the SSL record format:



TLS (Transport Layer Security):

TLS is an internet standard version of SSL. The current version of TLS is very similar to the SSL v3 with slight differences:

1. Version number – The TLS record format is similar to SSL record format. The header fields also have the same meaning. But the major version is 3 and minor version is 1for TLS.
2. The MAC value for TLS is also calculated in different. Here we use HMAC algorithm. It is defined as follows:

$$HMAC = H\,[(K^{+} \oplus Opad) \,||\, H\,[(K^{+} \oplus ipad) \,||\, M]]$$

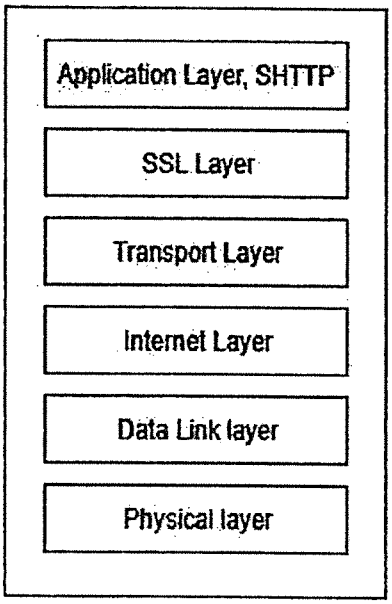
Where H – embedded hash function for TLS, SHA-1 or MD5 is used

M – message I / P to HMAC

K^{+} - secret key padded with 0s on left for MD% & SHA -1. Length of K^{+} is 512

Secure Hyper Text Transfer Protocol (SHTTP):

The SHTTP is a set of security of mechanisms defined for protecting the internet traffic. This includes data entry forms and Internet based transactions. Each S-HTTP file is either encrypted, contains a digital certificate, or both. A major difference is that S-HTTP allows the client to send a certificate to authenticate the user whereas, using SSL, only the server can be authenticated. The difference is shown in the following figure:

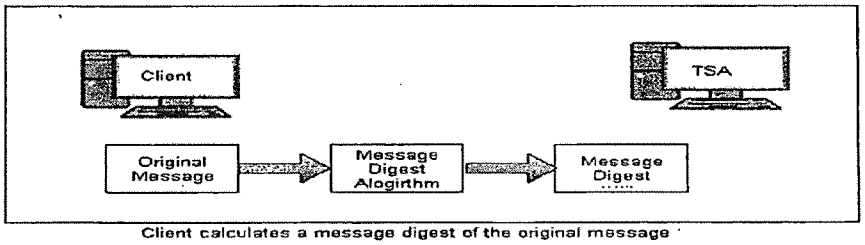


Positions of SHTTP and SSL in TCP/IP protocol suite

Time Stamping Protocol (TSP):

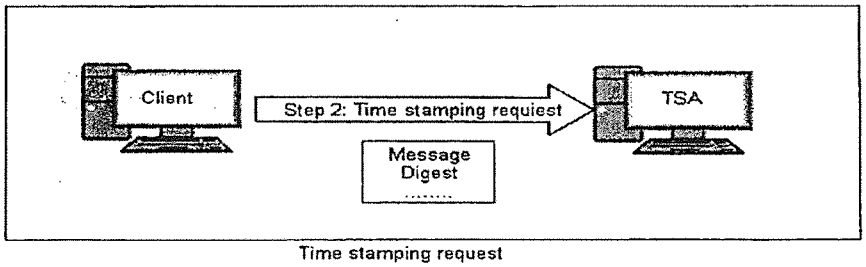
This provides a proof that a certain piece of data existed at a particular time. This Public Key Infrastructure (PKI) service is provided by an authority called as Time Stamping Authority (TSA). TSP is currently under the development of the PKIX working group. The TSP is a simple request response protocol, similar to HTTP. This work as described below, step-by-step:

Step 1: Message digest calculation: Firstly, the entry (client) requiring a timestamp calculates a message digest of the original message, which needs a timestamp from the TSA. The client should a use a standard message digest algorithm, such as MD5 or SHA-1 for this purpose. This is shown in the following figure:



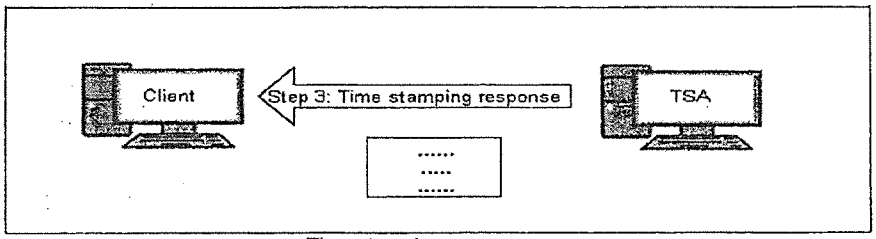
Client calculates a message digest of the original message

Step 2: Timestamping request: Now, the client sends the message digest calculated in the step 1 to the Time Stamp Authority (TSA) for getting a Timestamped and is known as Time Stamping Request. This is shown in the following figure:



Time stamping request

Step 3: Time stamping response: In response to the client's request, The TSA might decide to grant or reject the timestamp. If it decides to accept the request and process it, it signs the client's request together with the timestamp by the TSA private key. Regardless, it returns a Time Stamping Response back to the client. This is shown in the following figure:



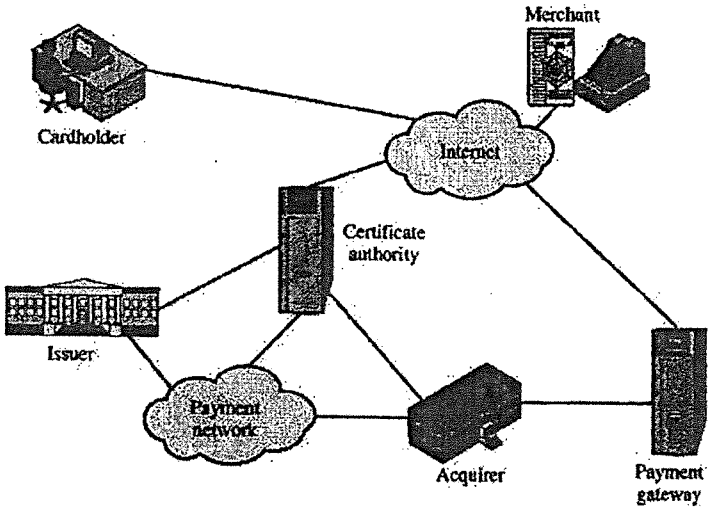
Time stamping response

Secure Electronic Transaction (SET):

The SET is an open encryption and security specification that is designed for protecting credit card transactions on the Internet. The current version, SETv1, emerged from a call for security standards by MasterCard and Visa in February 1996. A wide range of companies were involved in developing the initial specification, including IBM, Microsoft, Netscape, RSA, Terisa, and Verisign. SET provides three services:

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

SET Participants (Components):



The above Figure indicates the participants in the SET system, which include the following:

1. **Cardholder:** In the electronic environment, consumers and corporate purchasers interact with merchants from personal computers over the Internet. A cardholder is an authorized holder of a payment card (e.g., MasterCard, Visa) that has been issued by an issuer.
2. **Merchant:** A merchant is a person or organization that has goods or services to sell to the cardholder. Typically, these goods and services are offered via a Web site or by electronic mail. A merchant that accepts payment cards must have a relationship with an acquirer.

3. **Issuer:** This is a financial institution, such as a bank, that provides the cardholder with the payment card. Typically, accounts are applied for and opened by mail or in person. Ultimately, it is the issuer that is responsible for the payment of the debt of the cardholder.
4. **Acquirer:** This is a financial institution that establishes an account with a merchant and processes payment card authorizations and payments. Merchants will usually accept more than one credit card brand but do not want to deal with multiple bankcard associations or with multiple individual issuers. The acquirer provides authorization to the merchant that given cards account is active and that the proposed purchase does not exceed the credit limit. The acquirer also provides electronic transfer of payments to the merchant's account. Subsequently, the acquirer is reimbursed by the issuer over some sort of payment network for electronic funds transfer.
5. **Payment gateway:** This is a function operated by the acquirer or a designated third party that processes merchant payment messages. The payment gateway interfaces between SET and the existing bankcard payment networks for authorization and payment functions. The merchant exchanges SET messages with the payment gateway over the Internet, while the payment gateway has some direct or network connection to the acquirer's financial processing system.
6. **Certification authority (CA):** This is an entity that is trusted to issue X.509v3 public-key certificates for cardholders, merchants, and payment gateways. The success of SET will depend on the existence of a CA infrastructure available for this purpose. As was discussed in previous chapters, a hierarchy of CAs is used, so that participants need not be directly certified by a root authority.

SET Transaction:

We now briefly describe the sequence of events that are required for a transaction.

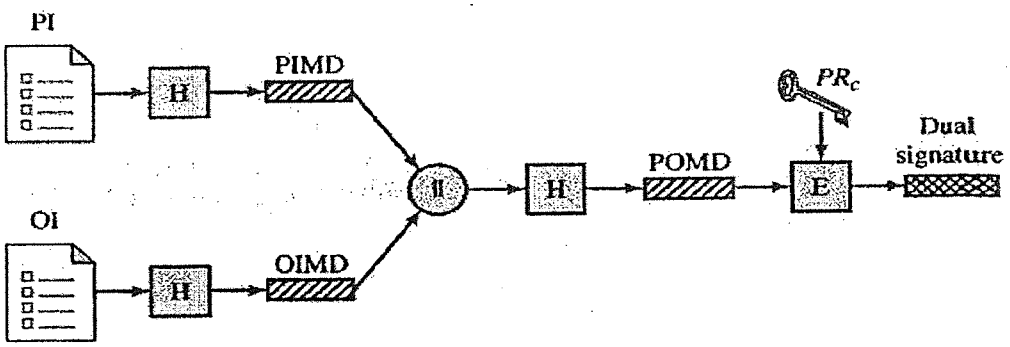
1. **The customer opens an account.** The customer obtains a credit card account, such as MasterCard or Visa, with a bank that supports electronic payment and SET.
2. **The customer receives a certificate.** After suitable verification of identity, the customer receives an X.509v3 digital certificate, which is signed by the bank. The certificate verifies the customer's RSA public key and its expiration date. It also establishes a relationship, guaranteed by the bank, between the customer's key pair and his or her credit card.
3. **Merchants have their own certificates.** A merchant who accepts a certain brand of card must be in possession of two certificates for two public keys owned by the

merchant: one for signing messages, and one for key exchange. The merchant also needs a copy of the payment gateway's public-key certificate.

4. **The customer places an order.** This is a process that may involve the customer first browsing through the merchant's Web site to select items and determine the price. The customer then sends a list of the items to be purchased to the merchant, who returns an order form containing the list of items, their price, a total price, and an order number.
5. **The merchant is verified.** In addition to the order form, the merchant sends a copy of its certificate, so that the customer can verify that he or she is dealing with a valid store.
6. **The order and payment are sent.** The customer sends both order and payment information to the merchant, along with the customer's certificate. The order confirms the purchase of the items in the order form. The payment contains credit card details. The payment information is encrypted in such a way that it cannot be read by the merchant. The customer's certificate enables the merchant to verify the customer.
7. **The merchant requests payment authorization.** The merchant sends the payment information to the payment gateway, requesting authorization that the customer's available credit is sufficient for this purchase.
8. **The merchant confirms the order.** The merchant sends confirmation of the order to the customer.
9. **The merchant provides the goods or service.** The merchant ships the goods or provides the service to the customer.
10. **The merchant requests payment.** This request is sent to the payment gateway, which handles all of the payment processing.

Dual Signature:

The purpose of the dual signature is to link two messages that are intended for two different recipients. In this case, the customer wants to send the order information (OI) to the merchant and the payment information (PI) to the bank. The merchant does not need to know the customer's credit card number, and the bank does not need to know the details of the customer's order. The following figure explains construction Dual Signature:



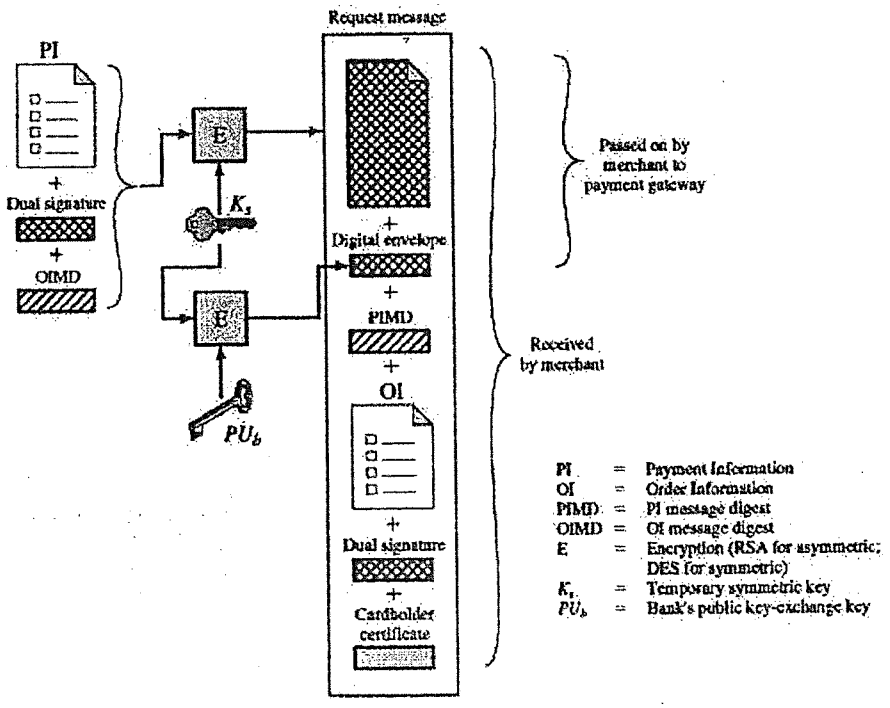
PI = Payment information
OI = Order information
H = Hash function (SHA-1)
|| = Concatenation

PIMD = PI message digest
OIMD = OI message digest
POMD = Payment order message digest
E = Encryption (RSA)
PR_c = Customer's private signature key

Purchase Request – Customer:

Before the Purchase Request exchange begins, the cardholder has completed browsing, selecting, and ordering. The end of this preliminary phase occurs when the merchant sends a completed order form to the customer. All of the preceding occurs without the use of SET.

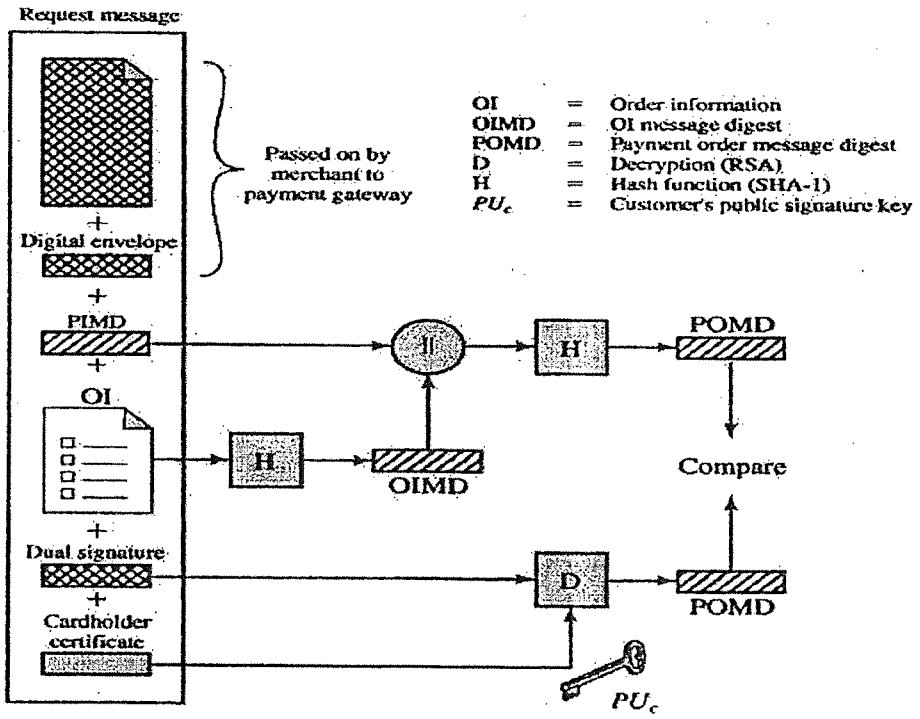
The purchase request exchange consists of four messages: Initiate Request, Initiate Response, Purchase Request, and Purchase Response. This is explained in the following figure:



Cardholder Sends Purchase Request

Purchase Request – Merchant:

1. verifies cardholder certificates using CA sigs
2. verifies dual signature using customer's public signature key to ensure order has not been tampered with in transit & that it was signed using cardholder's private signature key
3. processes order and forwards the payment information to the payment gateway for authorization (described later)
4. sends a purchase response to cardholder



Merchant Verifies Customer Purchase Request

Payment Gateway Authorization:

1. verifies all certificates
2. decrypts digital envelope of authorization block to obtain symmetric key & then decrypts authorization block
3. verifies merchant's signature on authorization block
4. decrypts digital envelope of payment block to obtain symmetric key & then decrypts payment block
5. verifies dual signature on payment block

6. verifies that transaction ID received from merchant matches that in PI received (indirectly) from customer
7. requests & receives an authorization from issuer
8. sends authorization response back to merchant

Payment Capture:

1. merchant sends payment gateway a payment capture request
2. gateway checks request
3. then causes funds to be transferred to merchants account
4. notifies merchant using capture response

SSL versus SET:

The following table gives us an idea that SET is standard that describes a very complex authentication mechanism that makes it almost impossible for either party to commit any sort of fraud. But there is no such mechanism in SSL.

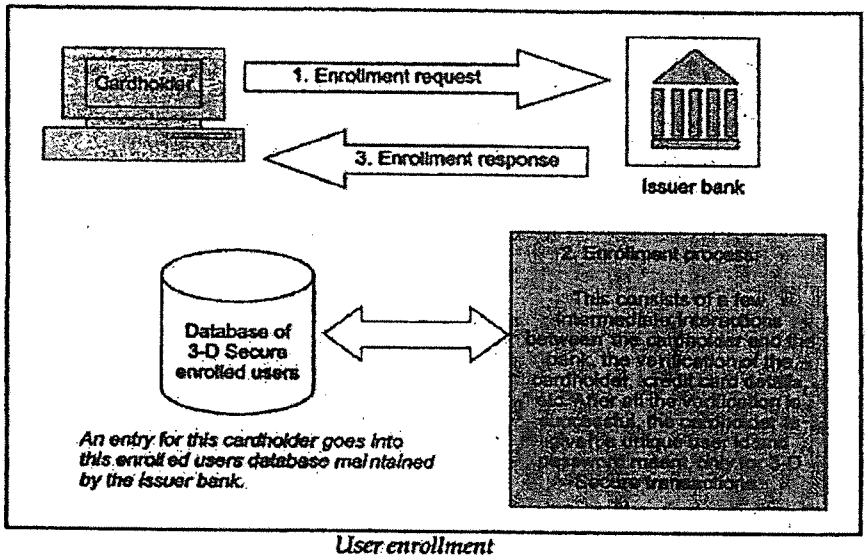
Issue	*SSL*	*SET*
Main aim	Exchange of data in an encrypted form	E-commerce related payment mechanism
Certification	Two parties exchange certificates	All the involved parties must be certified by a trusted third party
Authentication	Mechanisms in place, but not very strong	Strong mechanisms for authenticating all the parties involved
Risk of merchant fraud	Possible, since customer gives financial data to merchant	Unlikely, since customer gives financial data to payment gateway
Risk of customer fraud	Possible, no mechanisms exist if a customer refuses to pay later	Customer has to digitally sign payment instructions
Action in case of customer fraud	Merchant is liable	Payment gateway is liable
Practical usage	High	Not much

3-D Secure Protocol:

3-D Secure is an XML-based protocol designed to be an additional security layer for online credit and debit card transactions. It was originally developed by Arcot Systems (now Certificate Authority (CA) Technologies) and first deployed by Visa with the intention of improving the security of Internet payments and is offered to customers under the name Verified by Visa.

The main difference between SET and 3-D Secure is that any cardholder who wishes to participate in a payment transaction involving the use of the 3-D secure protocol has to enroll on the issuer bank's Enrollment Server. That is, the cardholder makes a card payment;

he/she must enroll with the issuer bank's Enrollment Server. This process is shown in the following figure:

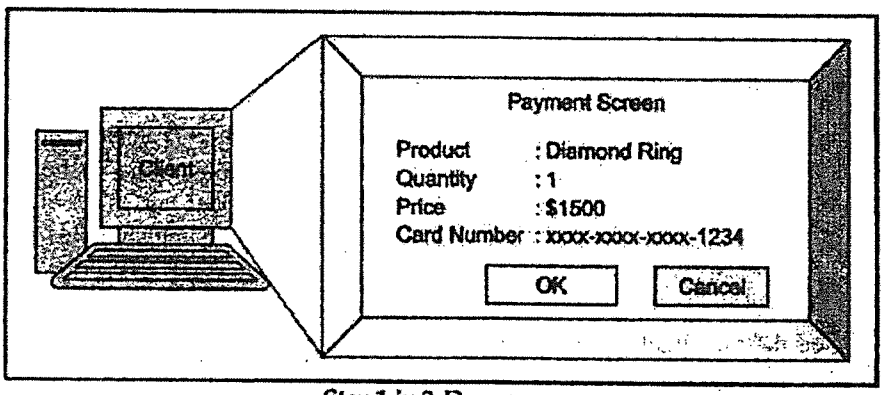


User enrollment

How 3-D Secure Protocol Works?

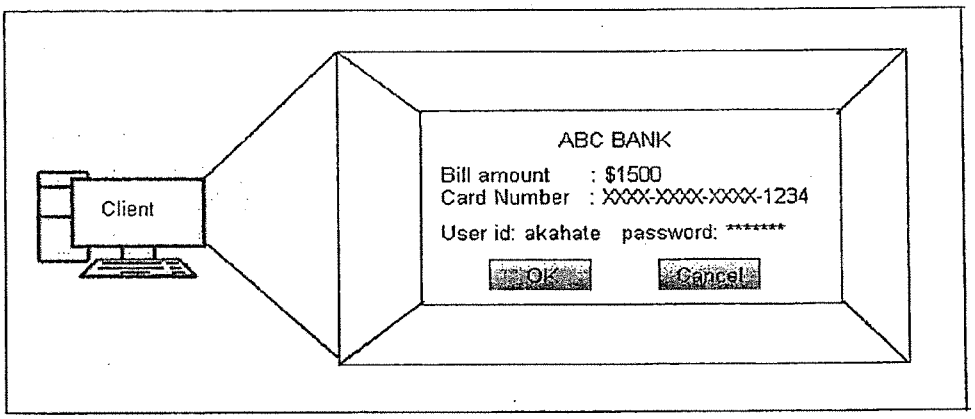
The following steps explain 3-D secure protocol works:

Step 1: The user shops using the shopping cart on the merchant site and decides to pay amount. The user enters the credit card details and clicks on OK button. This is shown in the following figure:



Step 1 in 3-D secure

Step 2: The user will be redirected to issuer's bank site and enters the password given by the bank. This is shown in the following figure:

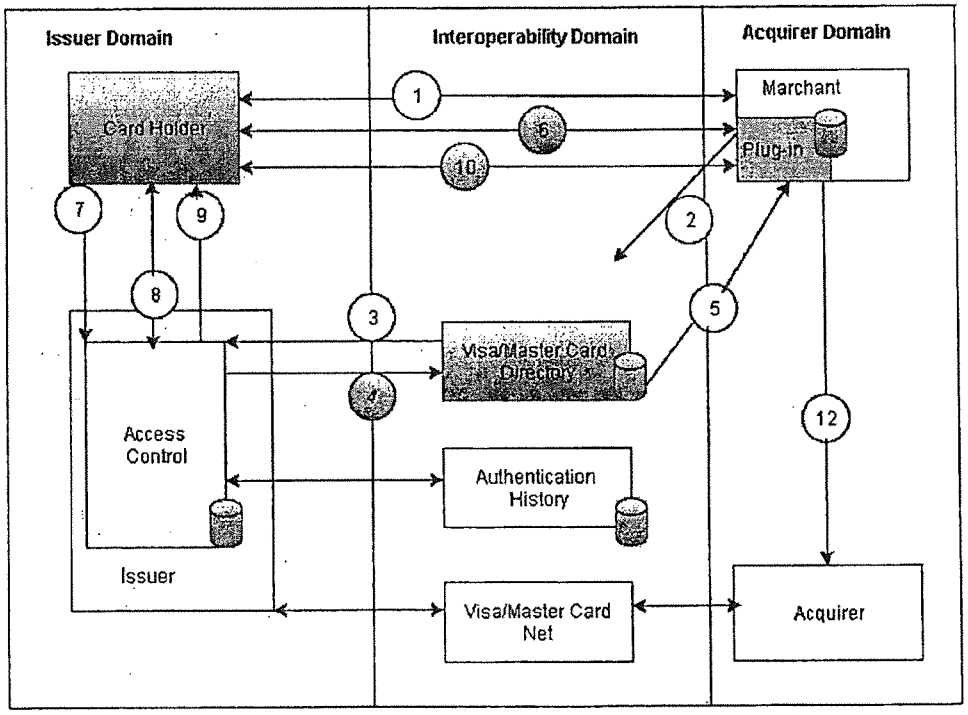


Step 2 in 3-D secure

At this stage, the bank verifies the user's password by comparing it with its database entry. The bank sends an appropriate success/failure message to the merchant and shows the corresponding screen to the user.

What Happens Behind the Scene?

The following figure describes the internal operations of 3-D Secure. The process uses SSL for confidentiality and server authentication.



3d Secure internal flow

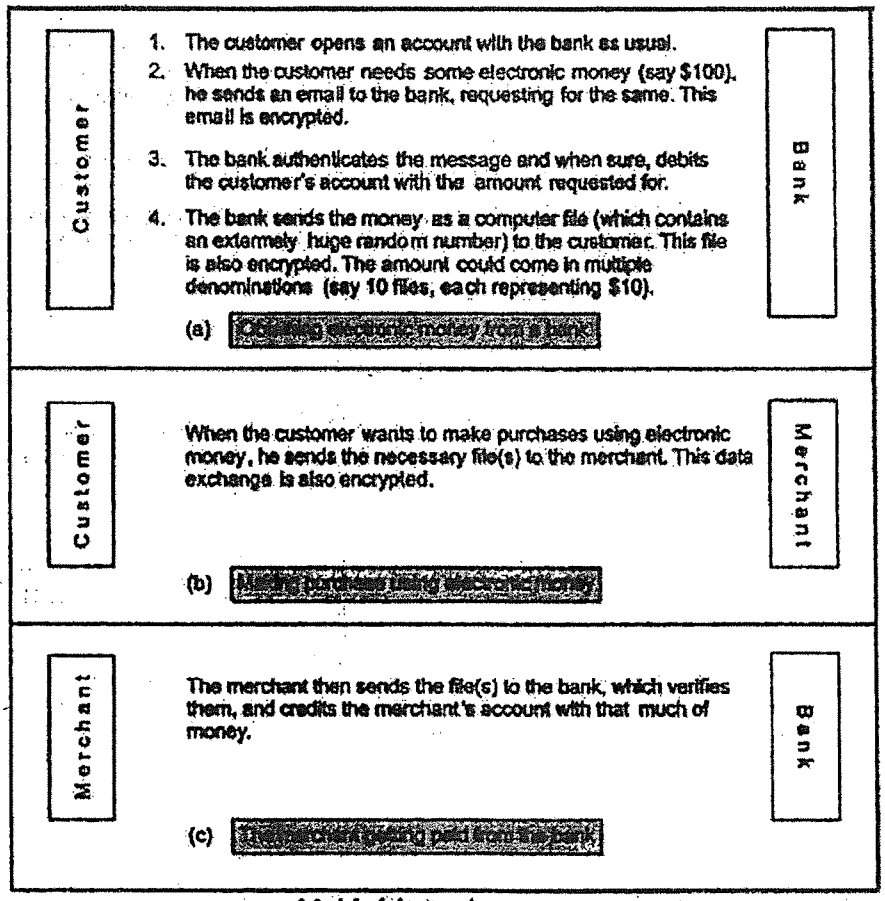
The flow can be described as follows:

1. The customer finalizes on the payment on merchant side (the merchant has all the data of this customer).

2. A program called as *merchant plug in*, which resides at the merchant Web server, sends the user information to the Visa/Master Card directory (which as LDAP-based).
3. The Visa/MasterCard directory queries access control server running at the issuer bank (i.e. the customer's bank), to check the authentication status of the customer.
4. The access control server forms the response for the Visa directory and sends it back to the Visa/MasterCard directory.
5. The Visa/MasterCard directory sends the payer's authentication status to the merchant plug in.
6. After getting response, if the user is currently not authenticated, the plug in redirects the user to the bank site, requesting the bank or the issuer site to perform the authentication process.
7. The access control server (running on the bank's site) receives the request for authentication of the user.
8. The authentication server performs authentication of the user based on the mechanism of authentication chosen by the user (e.g. password, dynamic password, mobile, etc.).
9. The access control server returns the user authentication information to the merchant plug in running in the acquirer domain by redirecting the user to the merchant site. It also sends the information to the repository where the history of the user authentication is kept for legal purpose.
10. The plug in receives the response of the access control server through the user's browser. This contains the digital signature of the access control server.
11. The plug in validates the digital signature of the response and the response from the access control server.
12. If the authentication was successful and the digital signature of the access control server is validated, the merchant sends the authorization information to its bank (i.e. the acquire bank).

Electronic Money:

Electronic money, which is also called as electronic cash or digital cash, is one more way of making payments on the Internet. Electronic money is nothing but money represented by computer files. In other words, the physical form of money is converted into binary form of computer data. The following figure shows the conceptual steps involved in electronic money processing:

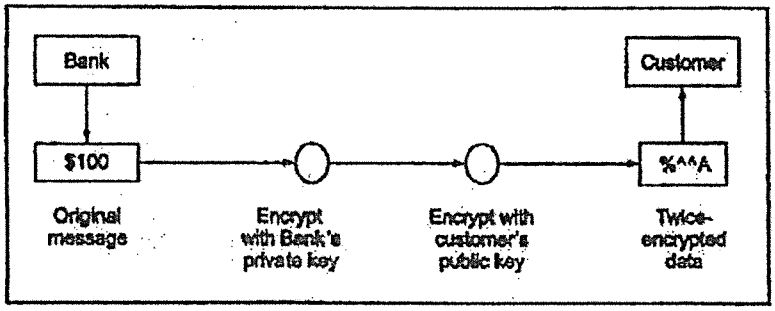


Model of electronic money

Security Mechanisms in Electronic Money:

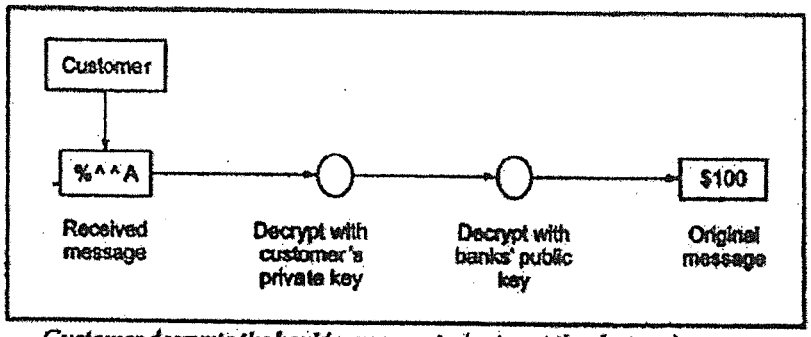
The following steps provide security mechanisms in Electronic Money.

Step 1: Bank sends the electronic money to the customer. This is shown in the following figure:



Bank sends electronic money to the customer after encrypting it twice

Step 2: The customer receives the money and decrypts it. This is shown in the following figure:



Customer decrypts the bank's message twice to get the electronic money

Types of Electronic Money:

Electronic money can be classified into two ways. They are

I. Classification based on the tracking of Money: This classification is based on whether the electronic money is tracked throughout its lifetime. Accordingly, it can be classified as follows:

1. Identified electronic money: This money more or less works like a credit card. The progress of the electronic money is identified by bank to one of its customer. All transactions of the customer are tracked by the bank and also it can identify whether the customer is original or not. The bank maintains a file for all transactions and each one is identified by unique serial number.

For example, the serial number generated by the bank for electronic money worth $100 is say SR100. If the customer wants to spend this money, the corresponding files are sent to the merchant. The merchant would go back to the bank to redeem the electronic money and get real money. At this point, the bank again has the electronic money with the serial number SR100. Therefore, it knows that the customer has bought something worth $100 from a specific merchant on a specific date. This is shown in the following figure:

2. Anonymous electronic money: This type of money is also called ***blinded money*** works like real hard cash. There is no more trace of how money was spent. The bank cannot create any serial number like in Identified electronic money. But the customer creates the serial number. The process of the customer generating random number is as follows:

i. The customer generates a random number by some mathematical algorithm. The customer then multiplies it by another huge number (called as the ***blinding factor***).
ii. The customer sends the resulting number, called as ***blinded number*** to the bank.
iii. The bank does not know about the original number of Step (i).
iv. Bank signs (i.e. encrypts) the blinded number and sends it back to the customer.

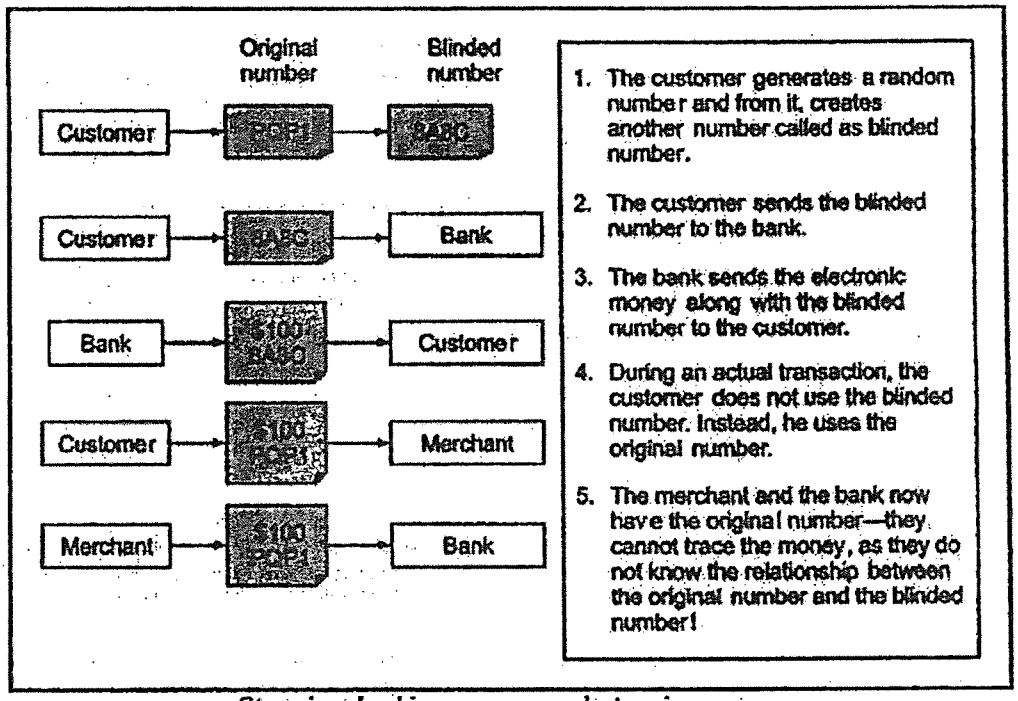
v. The customer converts the blinded number back to the original number using some algorithm.

vi. The customer then uses the original number (not the blinded number) when making any transaction with a merchant.

vii. The merchant's encashment request to the bank is also with the original number.

viii. The bank cannot trace this electronic money as it does not know the relationship between the original number and the blinded number.

This process is shown in the following figure:



Steps involved in anonymous electronic money

II. Classification Based on the Involvement of the Bank in the Transaction: Based on the involvement of the bank in the actual transaction electronic money can be further classified into two categories. They are

1. Online electronic money: In this type, the bank must actively participate in the transaction between the customer and the merchant. That is, before purchase transaction of the customer can complete, the merchant would confirm from the bank in real time as to whether the electronic money offered by the customer is acceptable (e.g. ensuring that it is not already spent, or that the serial number for it is valid).

2. Offline electronic money: In this type, the bank does not participate in the transaction between the customer and the merchant. That is, the customer purchases something from the merchant and offers to pay by electronic money. The merchant accepts the electronic money,

but does not validate it online. The merchant might collect a group of such electronic money transactions and process them together at a fixed time every day.

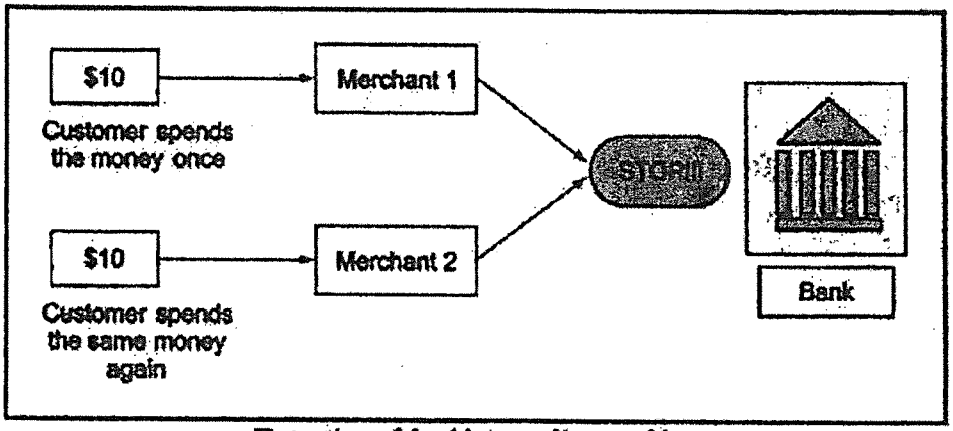
The Double Spending Problem:

If we combine the two ways of classifying electronic money, we have four possibilities:

1. Identified online electronic money
2. Identified offline electronic money
3. Anonymous online electronic money
4. Anonymous offline electronic money

Of the four, the last type can create the double spending problem. A customer could arrange for anonymous electronic money by using the blinded money concept. Later on, he could spend it offline more than once in quick succession (say in the same hour) with to different merchants. Since the bank is not involved in any of the two online transactions, the fact that same price of money is being spent cannot be prevented. Moreover, when it is realized that the same piece of money is spent more than once (when both merchants send their daily transaction lists to the bank), the bank cannot determine which customer spent it more than once, because of the blinding factor. Consequently, anonymous offline electronic money is of little practical use.

Double spending problem can happen in case of identified offline electronic money as well. However, upon detection, the customer under question can be easily tracked from the serial numbers of the electronic money. This is shown in the following figure:

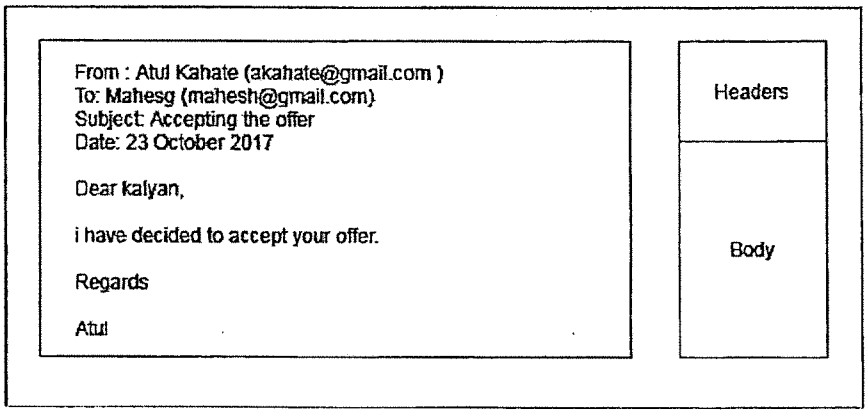


Detection of double spending problem

However, this detection is not possible in case of anonymous offline electronic money. Double spending problem is not possible either of the online transactions because the bank is a part of transaction between the customer and the merchant.

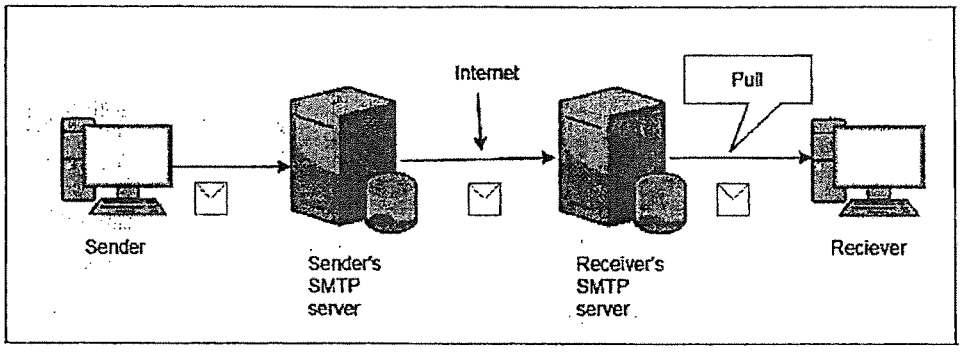
Email Security:

Electronic mail (email) is perhaps the most widely used application on the Internet. Email users can send and receive data easily on the Internet. Consequently, the security of email messages has become an extremely important issue. RFC 822 (Request for Comments) defines a format for test email messages. An email consists two parts: Headers and Body (contents). A header line consists of keyword, followed by a colon, followed by the keyword's arguments. Examples of header keywords are From, To, Subject and Date. The following figure distinguishes between its headers and contents.



Email headers and body

Simple Mail Transfer Protocol (SMTP): SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end. The following figure shows Email using the SMTP protocol:



Email using the SMPT protocol

The basic phases of an email communication consist of the following steps:

1. At the sender's end, an SMTP server takes the message sent by a user's computer.
2. The SMTP server at the sender's end then transfers the message to the SMTP server of the receiver.

3. The receiver's computer then pulls the email message from the SMTP server at the receiver's end, using email protocols such as Post Office Protocol (POP) or Internet Mail Access Protocol (IMAP).

SMTP is actually quite simple. The communication between a client and a server using SMTP consists of human-understandable ASCII text. The following steps describe the email communication using SMTP:

1. Based on the client's request for an email message transfer, the server sends back a *READY FOR MAIL* reply, indicating that it can accept an email message from the client.
2. The client then sends a *HELO* (abbreviation of *HELLO*) command to the server and identifies itself.
3. The server then sends back an acknowledgement in the form of its own DNS (Domain Name System) name.
4. The client can now send one or more email messages to the server. The email transfer begins with a MAIL command that identifies the sender.
5. The recipient allocates buffers to store the incoming email message and sends back an OK response to the client. The server also sends back a return code 250, which essentially means OK (human prefer OK and application programs prefer code 250).
6. The client now send email message to the recipient/recipients by using one or more RCPT commands.
7. After all RCPT commands, the client sends a DATA command, informing the server that the client is ready to start transmission of the email message.
8. The server responds back with a 354 Start email input message, indicating that it is ready to accept the email message. It also tells the client what identifiers it should send to signify that the message is over.
9. The client sends the email message and when it is over, sends the identifier provide by the server to indicate that its transmission is over.
10. The server sends back a 250 OK response.
11. The client sends a QUIT command to the server.
12. The server sends back a 221 Service closing transmission channel message, indicating that it is also closing its portion of the connection.

The actual interaction between the client and server as described above is shown in the following figure. The server has informed the client that it would like to see a

<CR><LF><LF> identifier when the client's transmission is over. Accordingly, the client sends this when its transmission is over.

```
S: 220 hotmail.com Simple Mail Transfer Service Ready
C: HELO yahoo.com
S: 250 hotmail.com

C: MAIL FROM: <Atul@yahoo.com>
S: 250 OK

C: RCPT TO: <Ana@hotmail.com>
S: 250 OK

C: RCPT TO: <Jul@hotmail.com>
S: 250 OK

C: DATA
S: 354 Start mail input; end with <CR><LF><LF>
C: … actual contents of the message …
C: …
C: …
C: <CR><LF><LF>
S: 250 OK

C: QUIT
S: 221 hotmail.com Service closing transmission channel
```

Example of an email message using SMTP protocol

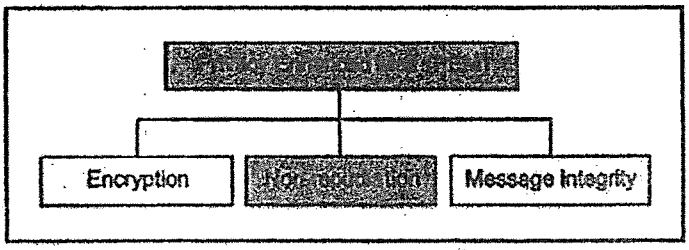
Email Security Protocols:

The following main three types of security protocols are providing security to the emails.

1. Privacy Enhanced Mail (*PEM*)
2. Pretty Good Privacy (*PGP*)
3. Secure and Multipurpose Internet Mail Extension (*S/MIME*)

PRIVACY-ENHANCED ELECTRONIC MAIL (PEM):

The Security is provided at the Application Layer using PEM. PEM was developed by IETF (Internet Engineering Task Force). PEM supports three main cryptographic functions which are shown in the following figure:



Security features offered by PEM

PEM Basic Design:

1) Defines two keys
   a) Data Encipherment Key (DEK) to encipher the message sent
      i) Generated randomly
      ii) Used only once
      iii) Sent to the recipient
   b) Interchange key: to encipher DEK
      i) Must be obtained some other way than the through the message

Protocols:

1) Confidential message (DEK: k_s)

$$\text{Alice} \xrightarrow{\{m\}k_s \,\|\, \{k_s\}k_{Bob}} \text{Bob}$$

2) Authenticated, integrity-checked message

$$\text{Alice} \xrightarrow{m \,\|\, \{h(m)\}k_{Alice}} \text{Bob}$$

3) Enciphered, authenticated, integrity checked message

$$\text{Alice} \xrightarrow{\{m\}k_s \,\|\, \{h(m)\}k_{Alice} \,\|\, \{k_s\}k_{Bob}} \text{Bob}$$

The PEM specification consists of four parts:

1. Message format (in addition to RFC 2822)
2. Certificates (as defined in X.509), a certification authority (CA) hierarchy, and Certificate Revocation Lists (CRLs)
3. Cryptographic algorithms to be used (e.g. DES-CBC, RSA, MD5, …)
4. Formats of control messages

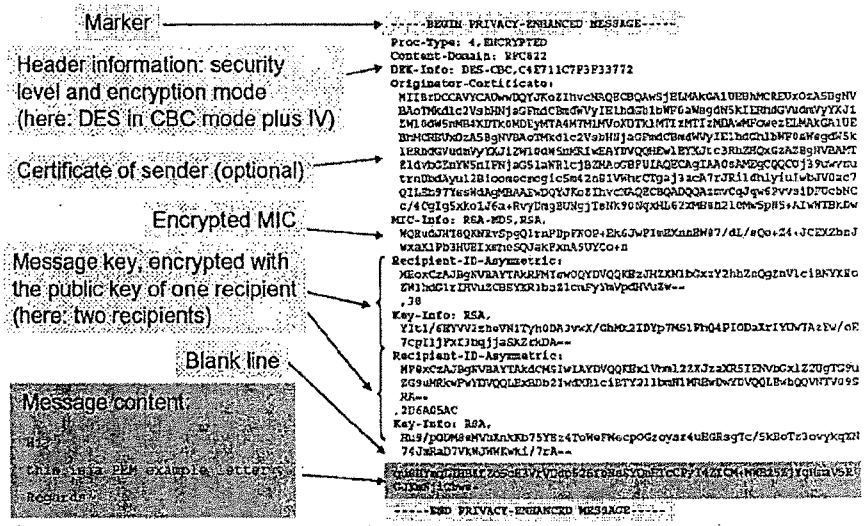
PEM Message Format:

PEM messages can consist of several parts of different security level:

1. *Ordinary*: unsecured data
2. *Integrity-protected unmodified data (MIC-CLEAR)*: the original message is included unmodified as part of the PEM message, but an integrity check is added
   a. *Integrity-protected encoded data (MIC-ONLY)*: the message first is base64-encoded, then an integrity check is added
3. *Encoded encrypted integrity-protected data (ENCRYPTED)*: an integrity check on the message is computed, then message and integrity check are encrypted with a randomly selected per-message secret key. The encrypted message/integrity check

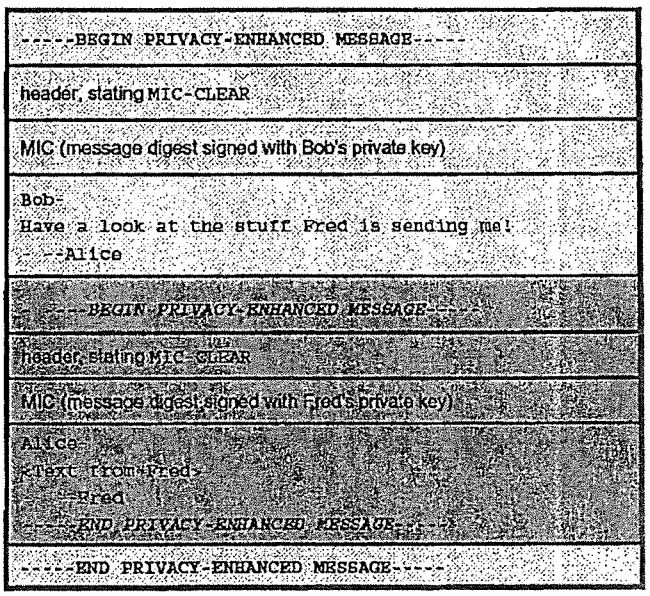
and the key are each base64-encoded to pass through message transfer agents as ordinary text. Parts are included in a normal mail body (like for MIME) and separated by markers.

PEM Message Example:



PEM - Forwarding and Enclosures:

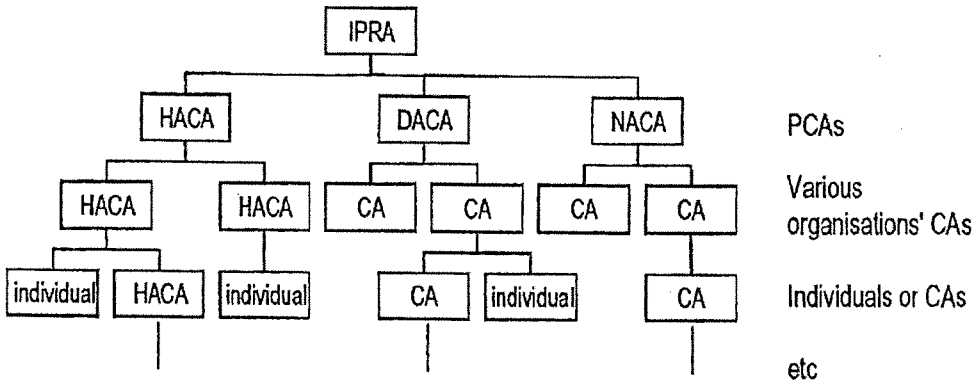
When Alice wants to forward Bob a message she received from someone else (say, Fred), she encapsulates it in a new message to keep Fred's signatures:



In case of a message in ENCRYPTED mode, Alice first has to decrypt the message from Fred and re-encrypt it with Bob's key.

PEM Certification Authority Hierarchy:

PEM specifies a single root CA called the IPRA (Internet Policy Registration Authority). IPRA certifies PCAs (Policy Certification Authorities), organized as a tree:



Each PCA has to enforce a policy from one of three possible security levels:

1. *High Assurance (HA)*: super-secure, i.e. implemented on special hardware, tamper resistant, etc.
2. *Discretionary Assurance (DA)*: well managed at top level, but does not impose any rules on the organizations to which CA certificates are granted
3. *No Assurance (NA)*: only constraint: not allowed to issue two certificates with same name

PRETTY GOOD PRIVACY (PGP):

Pretty Good Privacy (PGP) is high-security cryptographic software application, which allows people to exchange messages or file with privacy (confidentially), authentication and integrity. PGP can also be used to encrypt and apply digital signature for e-mail, PGP was developed by Zimmermann in the 1980s and first version was released on Internet in 1991. Because of legal issues for usage of RSA, it was purchased by Via-Crypt and RSA licensed company in 1993 and released again in 1994.

There are number of reasons that make the PGP to use widely. Some of them are:

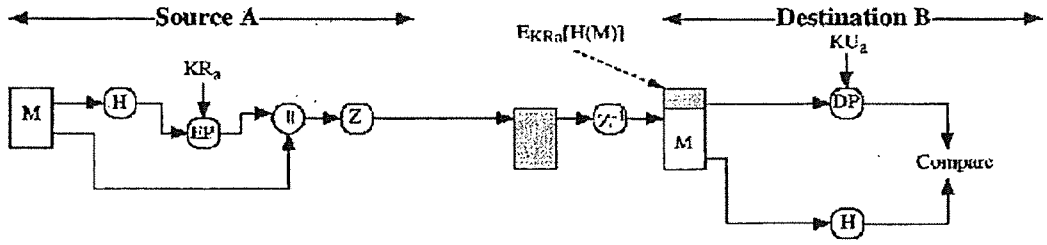
1. PGP is freeware, but also commercial versions available.
2. Operating system independent and run on Windows/Unix/Macintosh etc.
3. Used popular and standard algorithms like RSA, DSS, IDEA
4. Wide range of applications.
5. One of the major reasons is it was not controlled by any governmental or standard organization.

Pretty Good Privacy Operation:

The following table shows the services provided by the PGP along with common algorithms used in them.

Service	Function	Algorithm Used
Authentication →	Digital Signature	DSS/SHA or RSA/SHA
Confidentiality →	Message Encryption	CAST or IDEA or three-key triple DES with Diffie-Hellman or RSA
Speed →	Compression	ZIP
Transmission Size →	E-mail Compatibility	Radix-64 conversion
Limitations →	Segmentation	

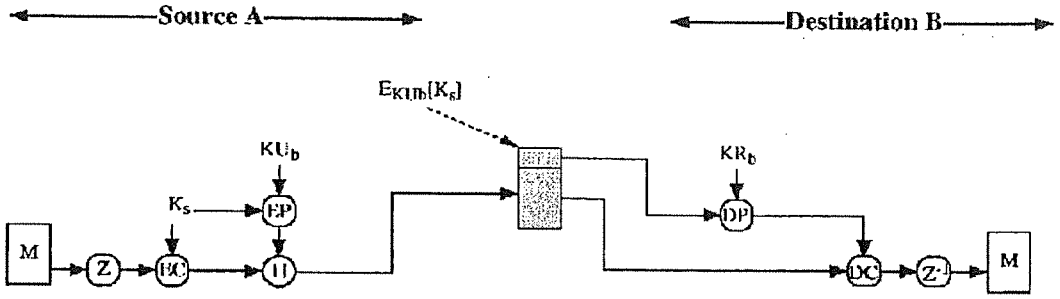
Authentication: The following diagram shows the authentication provision by using digital structure:



The above said scheme is called as digital signature scheme. Steps involved in the above diagram explained as follows:

1. At source machine, message is created.
2. Generation of 160-bit hash code of message by using SHA-1.
3. RSA algorithm is used for encrypting hash code; sender's private key is used for encryption so that authentication is provided. This hash code is used to the message.
4. Receiver decrypts the message by using public key of sender and recovers the hash code.
5. Now receiver generates hash code for the message and compares it with the decrypted hash code, if both are same then the message is authenticated.

Confidentiality: PGP provides basic service confidentiality. The following diagram shows how confidentiality is provided:

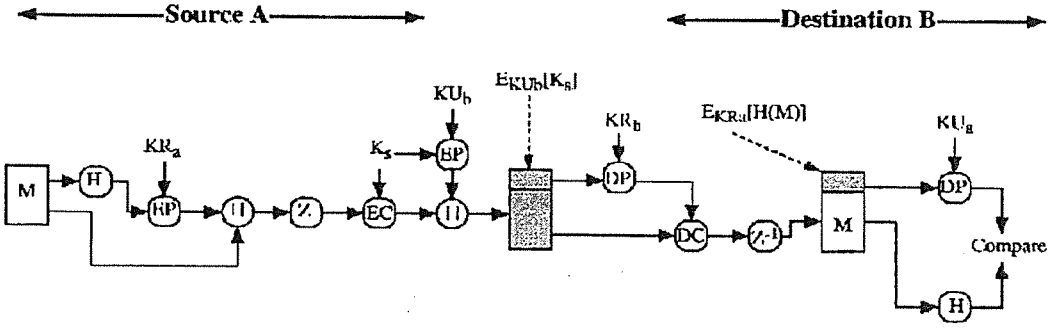


The process of confidentiality is described in the following steps.

1. At source machine message is created and a random 128-bit number is used as a session key.
2. By using symmetric encryption algorithm CAST-128 or IDEA or Tripled DES message is encrypted with the session key.
3. Confidentiality step taken place know, session key is encrypted with RSA by using the destination public key, and is added to the message.
4. Receiver decrypts the message by using private key of destination and recovers the session key.
5. Now receiver uses this session key to decrypt the message.

Confidentiality and Authentication:

To increase the trust of any service, we should provide both confidentiality and authentication. The following diagram shows both services:



In the above diagram the following steps are taken place.

1. At source machine message is created.
2. Generation of 160-bit hash code of the message by using SHA-1.

3. RSA algorithm is used for encrypting hash code; sender's private key is used for encryption so that authentication is provided. This hash code is used to the message.
4. A random 128-bit number is used as a session key.
5. By using symmetric encryption algorithm CAST-128 or IDEA or Triple DES signed message is encrypted with the session key.
6. Confidentiality step taken place know, session key is encrypted with RSA by using the destination public key, and is added to the message.
7. Receiver decrypts the signed message by using private key of destination and recovers the session key.
8. Now receiver uses this session key to decrypt the message.
9. Receiver decrypts the message by using public key of sender and recovers the hash code.
10. Now receiver generates hash code for the message and compares it with the decrypted hash code, if both are same then the message is authenticated.

Compression:

PGP compress the message after applying the signature but before encryption. The main reason behind it is:

1. Uncompressed message signing is better further verification.
2. Recompressed message verification is bit difficult because PGP's compression algorithm. It uses non-deterministic algorithm.

Message encryption is applied after compression (that reduces redundancy) automatically strengthens cryptographic security. Compression algorithm ZIP is used for compressing message in PGP.

Ex: Suppose the message is

"The brown fox jumped over the brown foxy jumping frog"

The message contains 53 characters. Here we have 424-bits in the message. The algorithm processes from left to right. Initially, we found that some characters are repeated i.e., "The brown fox is repeated after 26 characters and its length is 13 characters. So, we use another rotation for these 13 characters. Here we have two options for encoding:

01 – 12-bit pointer, 6-bit length

00 – 8-bit pointer, 4-bit length

The second occurrence of "The brown fox" is encoded in the following way.

$<00_b> <26_d> <13_d>$ i.e., 00 00011010 1101

At this point, the compressed message looks in the following way:

"The brown fox jumped over 00000110101101y jumping frog".

We also found that "_jump" occurs twice in the above sentence. So, the second "_jump" is encoded in the following way:

<00> <27_d> <5_d> i.e., 00 00011011 0101

Now the total message is compressed in the following way:

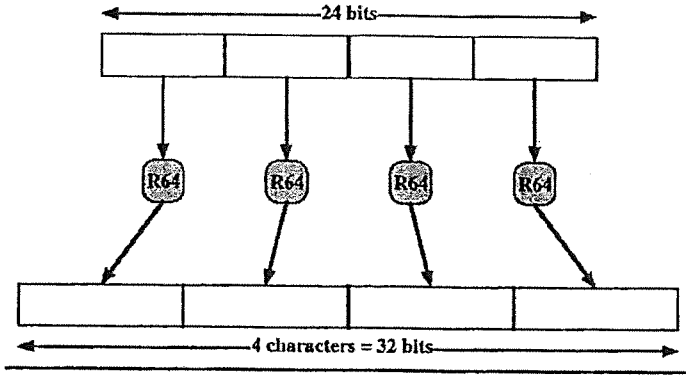
"The brown fox jumped over 00000110101101y00000110110101ing frog".

Its length is 35 characters, 28-bits i.e., a total of 28-bits. So, we observed that the compressed message is 116-bits less than the original message.

The compressed version is send to the destination after identifying the 14-bit code in the message. The receiver easily identifies the corresponding code.

E-mail Compatibility:

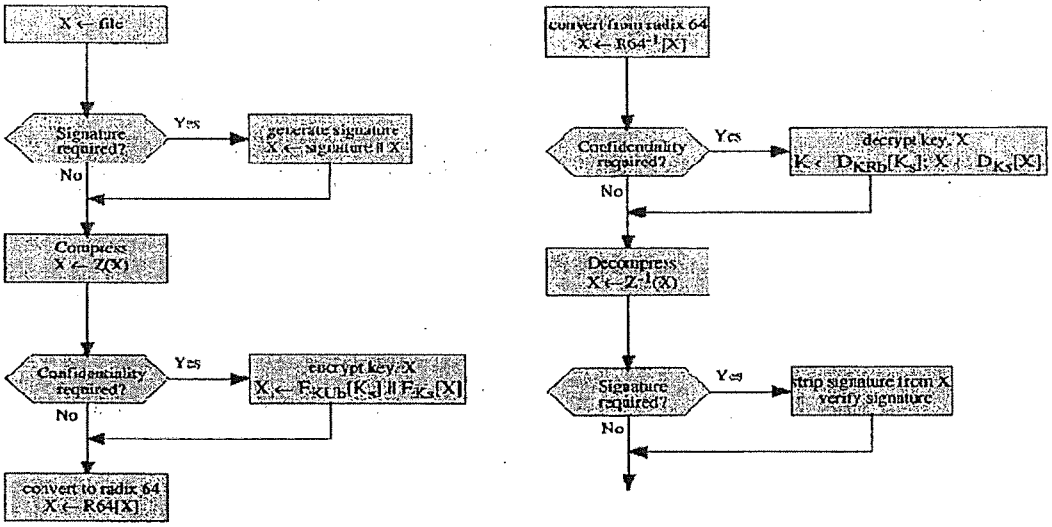
Radix-64 encoding conversion is E-mail compatible conversion. Each group of three octets of binary data (24 bits) is mapped into four ASCII characters (32 bits). It also adds cyclic redundancy check (CRC), useful for finding transmission errors. The following diagram shows this conversion process.



Segmentation and Reassembly:

E-mail systems impose maximum length on transmission. Some systems made this limit as 50 kb. For that reason PGP provides an automatic process that divides the message into small segments and this process is called segmentation. This was done after all other operations are over. Before transmitting the message segmentation takes place. At the receiving end reassembly of these segmented packets automatically done by PGP.

The following diagrams show all security services taken place at source and destination except segmentation and reassembly.

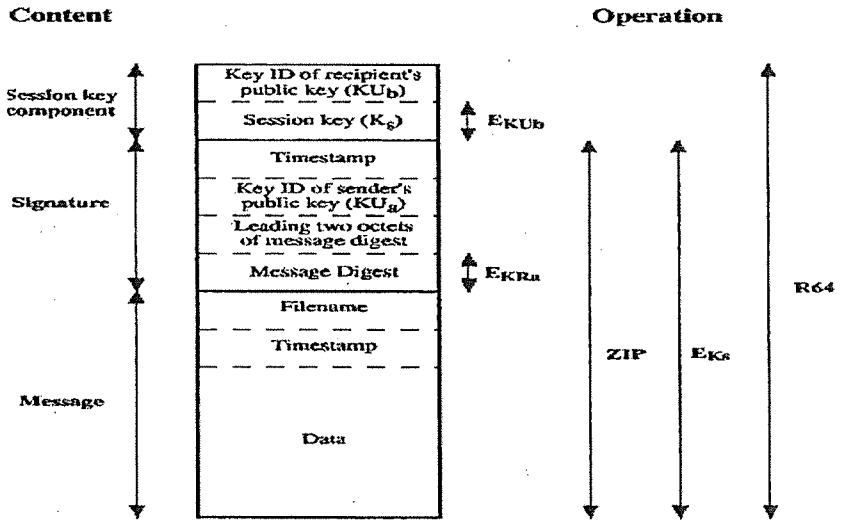


(a) Generic Transmission Diagram (from A)

(b) Generic Reception Diagram (to B)

Format of PGP Message:

The following diagram describes the format of PGP Message:



Key Management of PGP:

PGP uses the following four types of keys for its transmission:

1. One-time Session Symmetric Keys.
2. Public Keys (of Public Key Cryptography)
3. Private Keys (of Public key cryptography)
4. Pass phrase-based symmetric keys.

Key Rings: To overcome the difficulty of sending key identifications with every message PGP provides two types of key rings, one is private key ring and other one is public key ring.

Private Key ring includes time stamps, Key ID, public key, encrypted private key and user identification, Public Key ring includes time stamp, key id, public key, owner trust, user id, key legitimacy, signature and signature trust. Purpose of each field is as follows:

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
T_i	$KU_i \bmod 2^{64}$	KU_i	$E_{H(Pi)}[KR_i]$	User i
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.

Public Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
T_i	$KU_i \bmod 2^{64}$	KU_i	trust_flag$_i$	User i	trust_flag$_i$		
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

* = field used to index table

General Structure of Private and Public Key Rings

PGP Message Generation:

Sending steps for the PGP entity are as follows:

1. Signing the message.
2. Encrypting the message.

Receiving steps for the PGP entity are as follows:

1. Decrypting the message.
2. Authenticating (Verifying) the message.

Detailed Sending Steps: Signing the message essentially consists of the following steps:

a. Observer the step 1 of diagram, there PGP retrieves the sender's private key from the private-key ring using user identification as index. If user identification was not provided then first private key from the private key ring is retrieved.

b. In step 2 PGP asks the user for the pass phrase to recover the encrypted private key.

c. Step 3 is used to construct the signature component of the message.

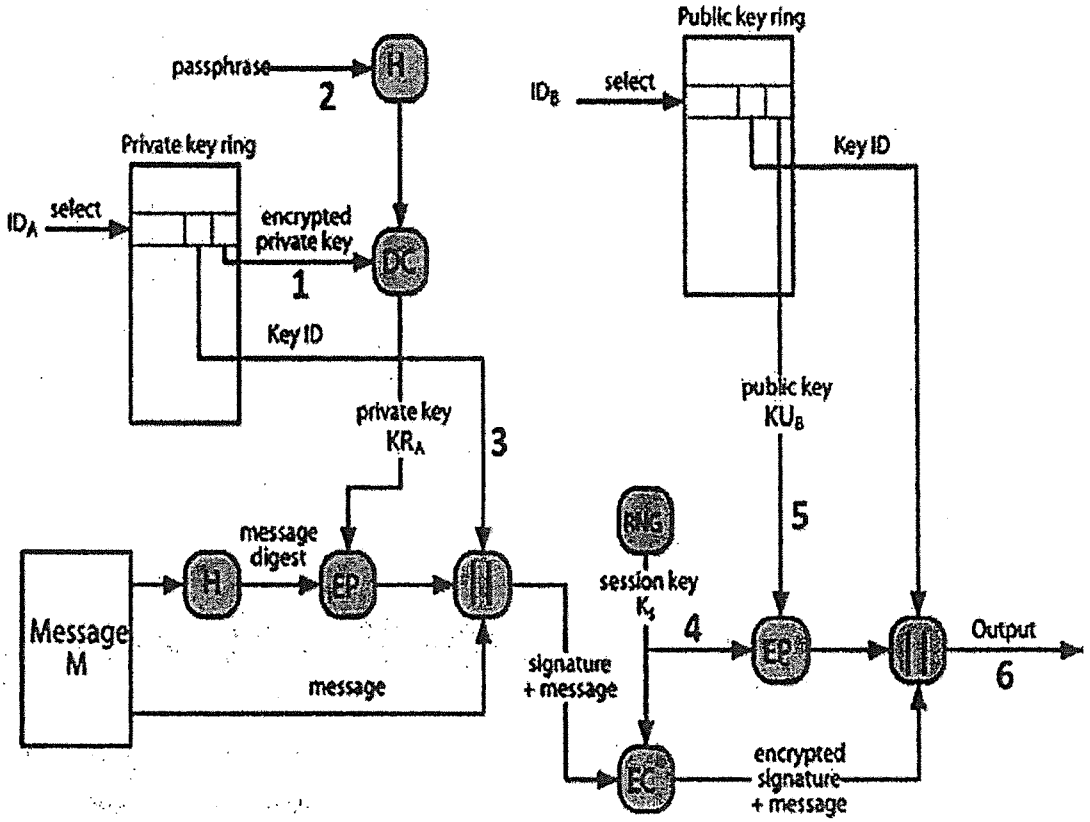
Encrypting the Message:

(i) In Step 4, PGP generates a session key and encrypts the message.

(ii) Observe step 5 here PGP retrieves the recipient's public key from the public key ring using user identification as an index.

(iii) Step 6 is used to construct the session key component of the message.

In the following diagram there are two blocks, one is signing block and other one is encrypting block. In signing and encrypting blocks the above steps are designed.



PGP Message Reception (Detailed Reception Steps):

Decrypting the Message:

(i) Observer the step 1 of diagram PGP retrieves the receiver's private key from the private key ring, using the key identification field in the session key component of the message as an index.

(ii) In step 2 PGP asks the user, for the pass phrase to recover the unencrypted private key.

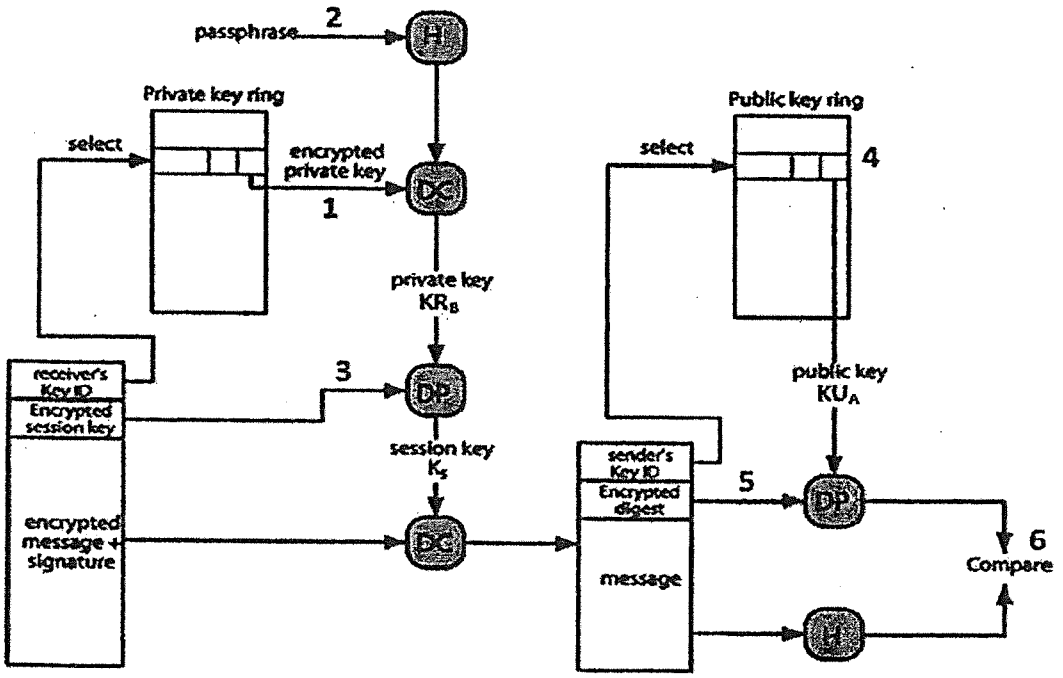
(iii) In step 3 PGP recovers the session key and decrypt the message.

Authenticating the Message:

(i) In Step 4, PGP retrieves the sender's public key from the public key ring, using the key identification field in the signature key component of the message as an index.

(ii) Observe step 5 here PGP recovers the transmitted Message Digest from the message.

(iii) In step 6 PGP computes the message digest for the received message and compares it to the transmitted message digest to authenticate the message.



SECURE/MULTIPURPOSE INTERNET MAIL EXTENSION (S/MIME):

S/MIME stands for Secure/Multipurpose Internal Mail Extensions; it is security enhance for S/MIME Internet email standard. Actual Internet RFC822 email can be able to transfer only text content only. MIME provides facility for transferring of varying content types and multi-part messages. It used encoding of binary data to textual form. S/MIME has support in various modern mail agents like MS Outlook, Netscape etc.

Multipurpose Internet Mail Extension (MIME):

Mainly MIME was developed to overcome the problems and limitations of the use of SMTP (Simple Mail Transfer Protocol), complete email system given in the first section of this chapter. Some of the limitations and problems of SMTP and some of them addressed by MIME are:

SMTP Cannot:

1. Transmit Executable files.
2. Transmit text data that contains Unicode characters or national language characters.
3. Transfer over a size limit.
4. Handle non-textual data included in X, 400 messages.

Common Problems are:

1. Handling of carriage return and linefeed characters.
2. Wrapping of lines.
3. Removal of trailing white space.
4. Padding of lines in a message to the same length.
5. Conversion of tabs into space characters.

Overview of MIME:

MIME specifications include five new messages that provide information about the body of the message, a number of content formats and transfer encoding techniques. The five header fields are MIME-Version, Content Type, Content Transfer Encoding technique, Content ID and Content Description. Purpose of each field is as follows:

1. **MIME Version:** Must have the value 1.0 indication of proper RFC.
2. **Content-Type** : Type of the content that is given in body is given here. Based on this information receiver user agent can pick an appropriate agent to represent data to the user.
3. **Content-Transfer-Encoding**: Type of transformation that is used to represent the body of the message in a way that is acceptable for mail transport.
4. **Content-ID**: It is used for identification of MIME entities.
5. **Content-Description:** Description of the object with the body. When you have audio kind of thing then this description is useful to know body content in detail.

MIME Content Types:

There are seven content types along with some sub types of them. The following table gives complete content types of MIME.

Type	Sub Type	Description
Text	Plain	Unformatted text
	Enriched	Formatted, rich text
Multipart	Mixed	Combination of different parts, but has some order
	Parallel	Same as Mixed but without any order
	Alternative	Different parts are alternative versions of the same
	Digest	Same as Mixed, but format varieties like msg/rfc
Message	Rfc822	Body is itself an encapsulated message that conforms to
	Partial	Fragmentation of large items
	External-body	A pointer reference to an external object
Image	Jpeg	JPEG format
	Gif	GIF format
Video	Mpeg	MPEG format
Audio	Basic	Single channel encoding technique
Application	Post Script	Adobe Post script format
	Octet-Stream	Normal binary data that consists of 8-bit bytes.

Generally content type declares the general type of data, and the subtype is used to specify a particular format for that type of data. There are two subtypes for the text type. One is plain and other one is enriched. Plain text is a string of ASCII characters, where as enriched text allows greater flexibility regarding format.

MIME Transfer Encodings:

Other major component of the MIME specification is definition of transfer encoding techniques. There are six different MIME transfer encoding techniques. The following table represents the MIME transfer encoding techniques.

Encoding	Description	Usage
7 bit	Short lines of ASCII	SMTP message transfer
8 bit	Short, but can have non-	Other mail transfer context
Binary	ASCII, non-ASCII and not	Other mail transfer context
Quoted-Printable	Human understandable form	Introduces non-safe characters
Base 64	Mapping of 6 bit blocks to 8-	Used in PGP
x-token	Named nonstandard encoding	Vendor-specific or application-

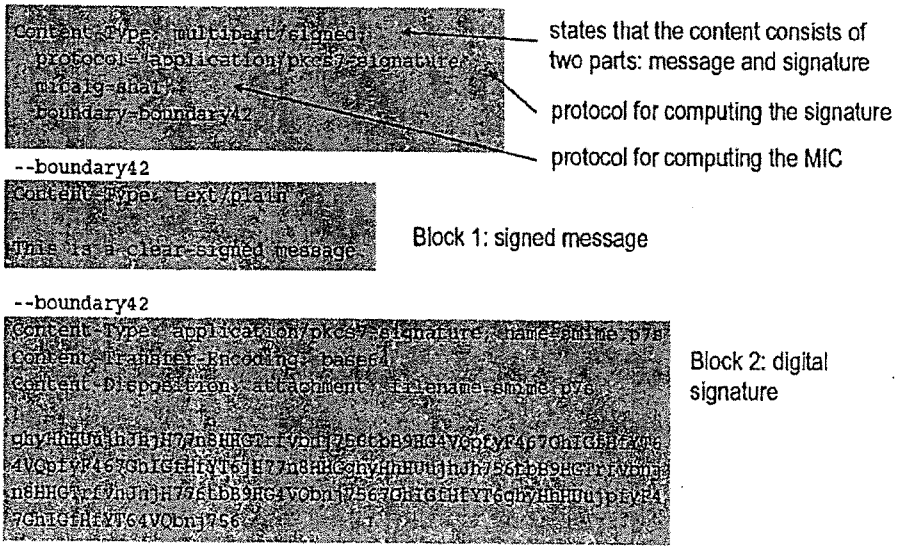
Functionality of S/MIME:

In terms of general functionality S/MIME is similar to PGP functionality. Digital signature, confidentiality and integrity services are provided by both algorithms. These two offer the ability to sign and/or encrypt the given messages. The following are functions of the S/MIME.

1. **S/MIME Enveloped Data:** The steps for preparing enveloped data MIME entity are as follows:
    a. Pseudorandom session key for a particular symmetric encryption algorithm was generated.
    b. Encrypt the session key
    c. Prepare public key certification with a block Recipient Info
    d. Encrypt the message content by using session.
2. **S/MIME Signed Data:** S/MIME signed data type gives integrity, authentication and non-repudiation services using sender signature. Multiple signers supported and prepare a signerInfo block for each one. Recipient checks signature using S/MIME entity embedded in public key system object. The following steps are used for preparing signed data MIME entity:
    a. First select message digest algorithm.
    b. Compute the message digest, or hash function for the message content.
    c. Encrypt Message Digest with the signer's private key.
    d. Private the public key certificate with a block SignerInfo.
3. **S/MIME Clear Signing:** By using multipart/signed type clear signing is achieved. Signing process does not involve transforming the message to be signed, so that the message is sent in clear format.
4. **S/MIME Registration Request:** We can use application/pkcs 10-MIME to transfer certification request. It includes certification RequestInfo block, identifier, and signature. The certification RequestInfo block includes a name of the certificate subject and user's public key.
5. **S/MIME Certificates:** Only Message: Certificates-only message is an application/ pkcs 7-MIMEtype/sub type with SMIME-type parameters of degenerate. Steps involved here are similar to step in creating signed Data message, except that there is no message content and signerInfo field is empty.

6. **S/MIME Certificate Processing:** S/MIME uses public key certificate that confirm to version 3 of X.509. The key-management scheme used by S/MIME was managed using a hybrid scheme of a strict X.509 certificate authority hierarchy and PGP's model of trust. Each client has a list of trusted certificate authority's certificates. Also it owns public or private key pairs and certificates. These certificates must be signed by trusted by certificate authorities. S/MIME user has several key-management functions to perform Key Generation, Registration and Certificate storage and retrieval.
7. **Enhanced Security Services:** There are three important security services that needs enhancement, they are:
    a. **Signed Receipts:** To confirm the mail is received by the recipient, there is a need of signed receipts. Main purpose of this furthers neither sender nor receiver can't deny the transmission.
    b. **Security Labels:** Security label is a set of security information regarding the sensitivity of the content. A security label may be included in the authenticate attributes of a signed object. Major use of security labels is access controls. Some other uses are priority of role-based.
    c. **Secure Mailing Lists:** There is a need of securing mailing list and also requires per-recipient processing Services of S/MIME Mail List Agent (MLA) send for this purpose. Always MLA takes single incoming message performs recipient-specific encryption.

S/MIME Example



Cryptographic Algorithms for S/MIME:

Service/Function	MUST	SHOULD
Message Digest used in	SHA1	SHA-1
Encrypt Digital Signatures	DSS	RSA
Session Key Encryption	Diffie-Hellman	RSA
Message Encryption	Decryption Triple DES	Encryption with Triple

In the above table second and third columns refers.

1. MUST indicate an absolute requirement of the specification.
2. SHOULD indicate not compulsory requirement but recommended.

S/MIME Messages:

As we have seen in the beginning of MIME, Secure MIME also supports all content types. In addition to that S/MIME provides some more content types. Some of the types are as follows:

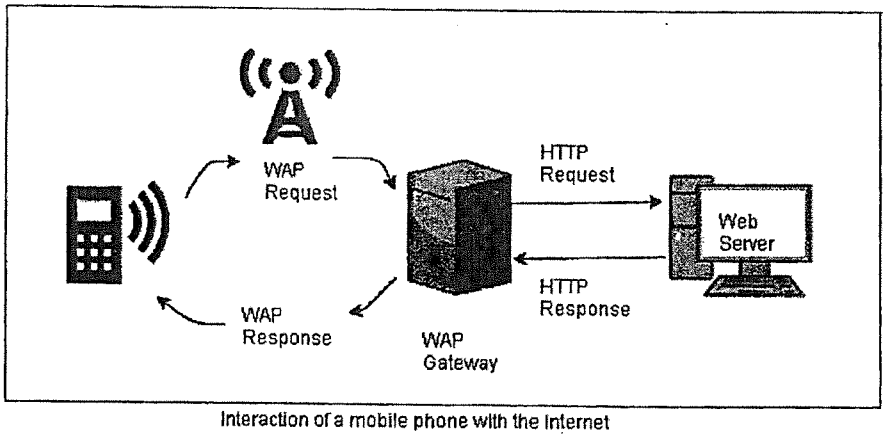
Type	Subtype	S/MIME parameter	Description
Multipart	Signed		Clear signed message
Application	Pkcs 7-mime	Signed Data	Signed S/MIME data
	Pkcs 7-mime	Enveloped Data	Encrypted S/MIME data
	Pkcs 7-mime	Degenerate Signed Data	Contains only public key
	Pkcs 7-		Content type of signature
	Pkcs 10-mime		Certificate registration

Wireless Application Protocol (WAP) Security:

Wireless Application Protocol commonly known as WAP is used to enable the access of internet in the mobile phones or PDAs. The main purpose of WAP is to enable easy, fast delivery of relevant information and services to mobile users. The important information of WAP is described as the following:

1. WAP is an application communication protocol
2. WAP is used to access services and information
3. WAP is for handheld devices such as mobile phones
4. WAP enables the creating of web applications for mobile devices.
5. WAP uses the Wireless Mark-up Language WML (not HTML) WML is defined as an XML 1.0 application

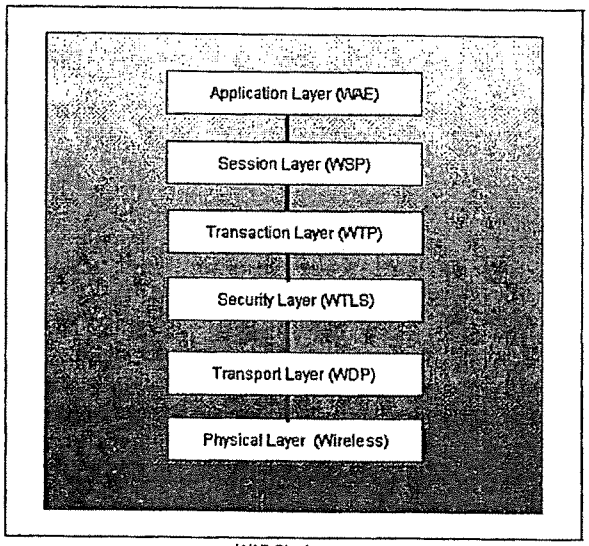
The WAP architecture has an additional level between the client and the server is *WAP gateway*. The WAP gateway translates client requests to the server form WAP to HTTP and way back from the server to the client, from HATTP to WAP. This is shown in the following figure:



Interaction of a mobile phone with the Internet

The WAP Stack:

The WAP Stack is based more on the OSI model, rather than the TCP/IP model. The structure of WAP stack is shown in the following figure:



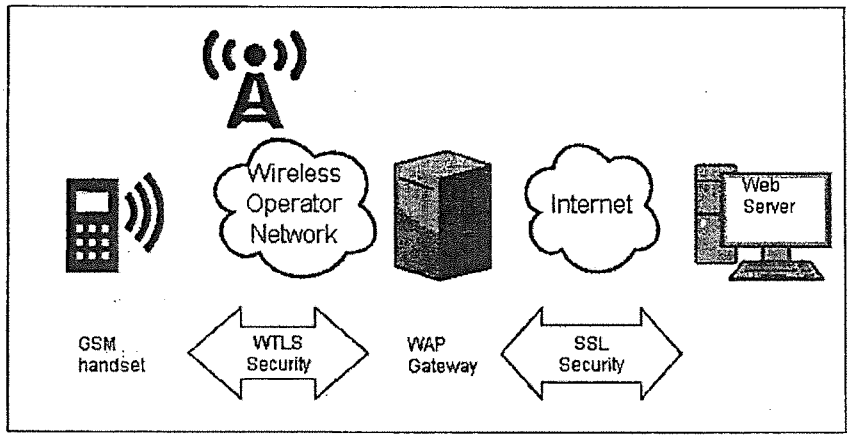
WAP Stack

The ***security layer*** of WAP stack is also called ***Wireless Transport Layer Security (WTLS)*** protocol. It is an optional layer, that when present, provides features such as authentication, privacy and secure connections – as required by many e-commerce and m-commerce (Mobile Commerce) applications. WTLS ensures the following four things:

1. **Privacy** ensures that the message passing between the client and the server are not accessible to anybody else.

2. **Server authentication** gives the client a confidence that the server is indeed what it is depicting as, and not someone who is posing as the server, with or without malicious intentions.
3. **Client authentication** gives the server a confidence that the client is indeed what is depicting as and not someone who is posing as the client, with or without malicious intentions.
4. **Data integrity** ensures that no one can tamper with the messages going between the client and the server, by modifying their contents in any manner.

The following figure shows how the communication between a WAP client and the origin server can be made secure. Between the WAP client and the WAP gateway, we have WTLS to ensure a secure-mode transaction. Between the WAP gateway and the origin server, SSL takes care of security, as usual. Thus the WAP gateway performs the transactions between WTLS and SSL in both directions.



WTLS and SSL Security

Security in GSM (Global System for Mobile Communication):

GSM is the most secured cellular telecommunications system available today. GSM has its security methods standardized. GSM maintains end-to-end security by retaining the confidentiality of calls and anonymity of the GSM subscriber. There three key aspects to GSM security:

1. Subscriber identity authentication
2. Signaling data confidentiality
3. User Data confidentiality

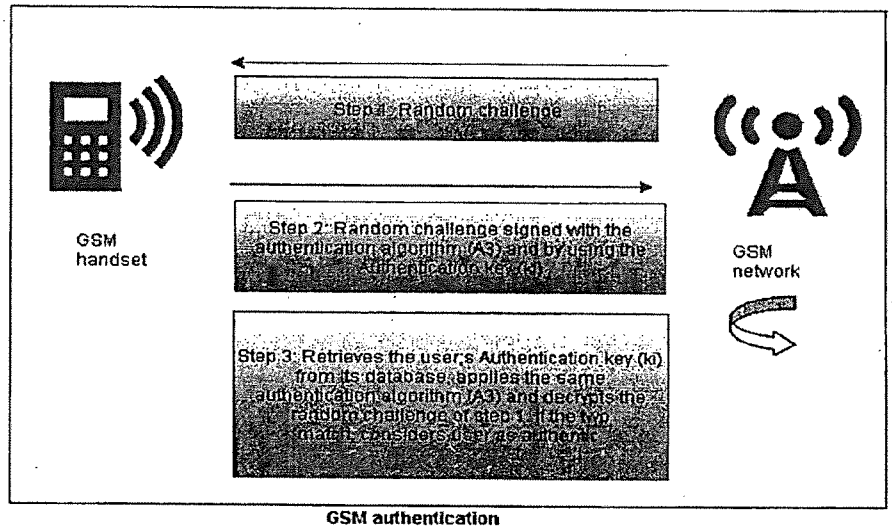
Each subscriber is identified with a unique International Mobile Subscriber Identity (IMSI). Each subscriber also has a unique subscriber authentication key (Ki). GSM

authentication and encryption work in such a way that this sensitive information is never transmitted across the mobile network. Instead, a challenge-response mechanism is used to perform authentication. The actual transmissions are encrypted with help of temporary, randomly generated ciphering key (Kc).

The security is distributed in three different elements of the GSM infrastructure: the Subscriber Identity Module (SIM), which is a plastic card inside a mobile phone, the GSM handset and the GSM network.

1. The SIM contains the IMSI, Ki, the ciphering key generation algorithm (A8), the authentication algorithm (A3) as well as Personal Identification Number (PIN).
2. The GSM handset contains ciphering algorithm (A5).
3. The Authentication Center (AUC), which is a part of GSM network, contains the encryption algorithm (A3, A5, and A8) as well as a database of identification and authentication information about the subscribers.

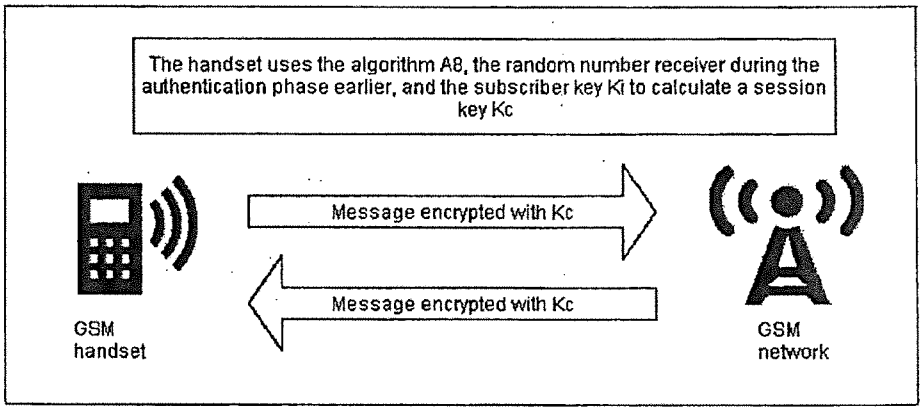
Authentication: The GSM authentication is shown in the following figure.



GSM authentication

The process begins with a challenge-response mechanism. The network sends a 128-bit random number to the subscriber when authentication begins. After this, 32-bit signed response using the authentication algorithm (A3) and the subscriber authentication key (Ki) is prepared by the handset and sent back to the network. The network retrieves its value of Ki from its database, performs the same operation using the A3 algorithm on the original 128-bit random number and compares this result with the one received from the handset. If the two match, the user is considered as successfully authenticated. Since the calculation of the signed response takes place inside the SIM, the IMSI or Ki never have to leave the SIM. That makes authentication secure.

Signaling and Data Confidentiality: The SIM contains the ciphering key generation algorithm (A8). This is used to produce the 64-bit ciphering key (Kc).The value of Kc is obtained by applying the same random number as used in authentication to the A8 algorithm with the individual subscriber authentication key (Ki). This key (Kc) is later used for secure communications between the subscriber and the mobile telephony base station. This process is shown in the following figure:



GSM encryption

Voice and Data Security: The A5 algorithm is used to encrypt the voice and data traffic between the user's handset and the GSM network. For this, the subscriber's handset sends a ciphering mode request to the GSM network. The network, in response, starts encryption and decryption of the traffic using ciphering algorithm (A5) and the ciphering key (Kc).

Note: The algorithms A3, A5, A8 are kept secret and are not available to the general public. However, they have been discovered, published on the Internet, and their implementations in C and other programming languages are available in many books/resources.

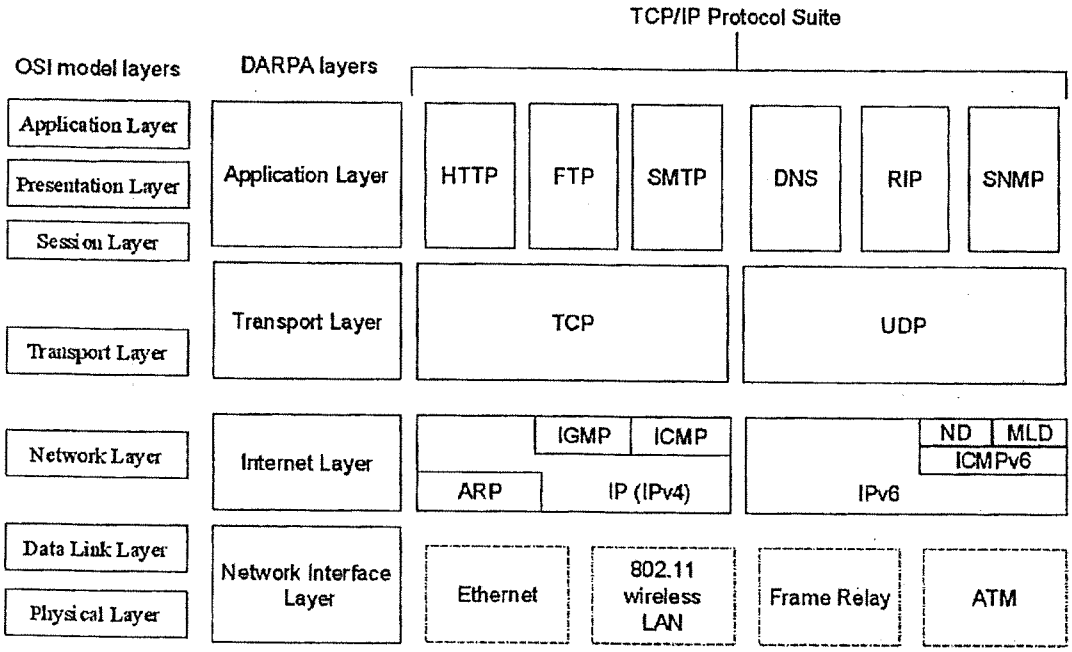
2. Network Security

TCP/IP:

Transmission Control Protocol/Internet Protocol (TCP/IP) is the protocol that is used on the Internet, which is the collection of thousands of networks worldwide that connect research facilities, universities, libraries, government agencies, private companies, and individuals.

The TCP/IP protocol suite maps to a four-layer conceptual model known as the DARPA model, which was named after the U.S. government agency that initially developed TCP/IP. The four layers of the Defense Advanced Research Projects Agency (DARPA) model are: Application, Transport, Internet, and Network Interface. Each layer in the DARPA model corresponds to one or more layers of the seven-layer OSI model. This model is shown in the following figure. The TCP/IP protocol suite has two sets of protocols at the Internet layer:

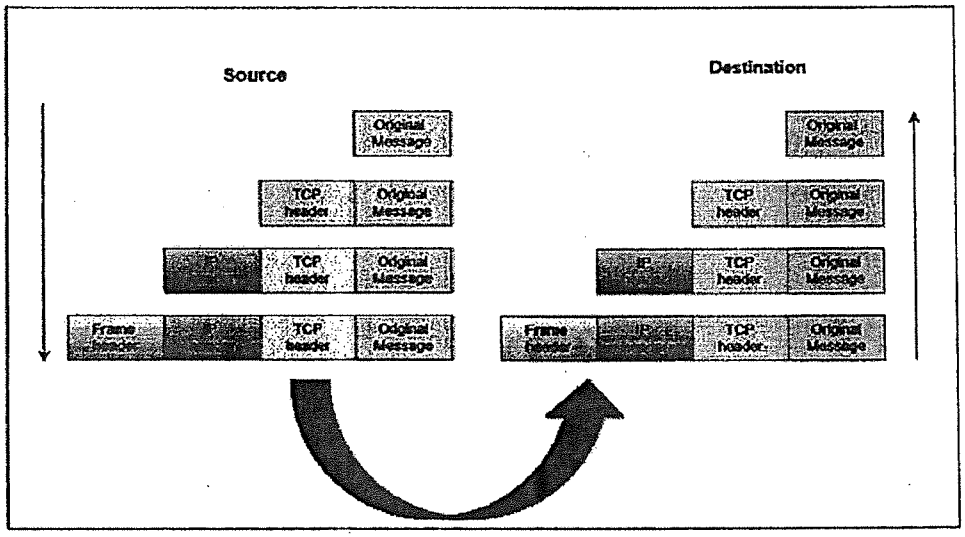
1. IPv4, also known as IP, is the Internet layer in common use today on private intranets and the Internet.
2. IPv6 is the new Internet layer that will eventually replace the existing IPv4 Internet layer.



Layers in the TCP/IP protocol suite

The data unit initially created at the application layer is called as a message. A message is actually broken down into segments by the transport layer. The network layer

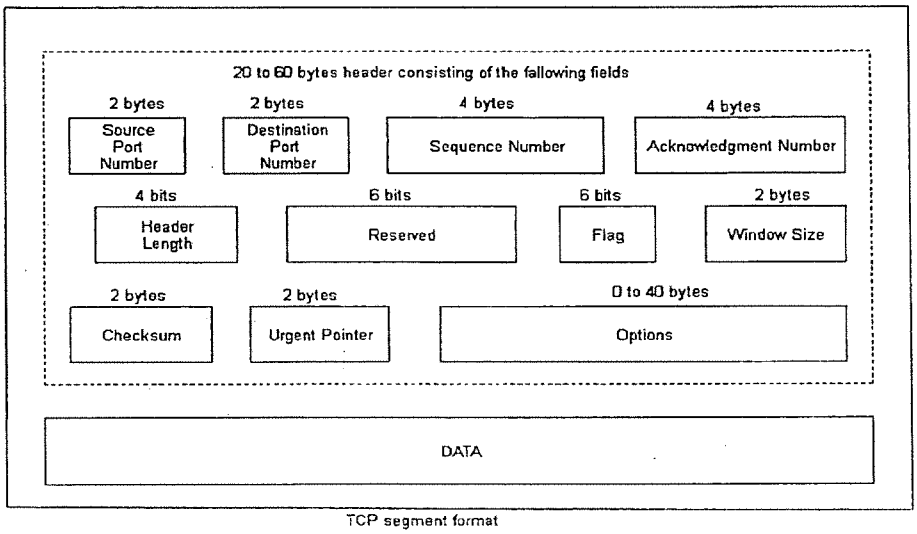
adds the IP header to this block and gives the result to the data link layer. The data link layer adds the frame header and gives it to the physical layer for transmission. At the physical layer, the actual bits transmitted as voltage pulses. At opposite process happens at the destination end, where each layer removes the previous layer's header and finally the application layer receives the original message. This is shown in the following figure:



Message transfer from the source to the destination at different TCP/IP layers

TCP Segment Format:

A TCP segment consists of a header of size 20 to 60 bytes, followed by the actual data. The header consists of 20 bytes if the TCP packet does not contain any options. Otherwise, the header consists of 60 bytes. That is, a maximum of 40 bytes are reserved for options. Options can be used to convey additional information to the destination. The TCP segment mat is shown in the following figure:



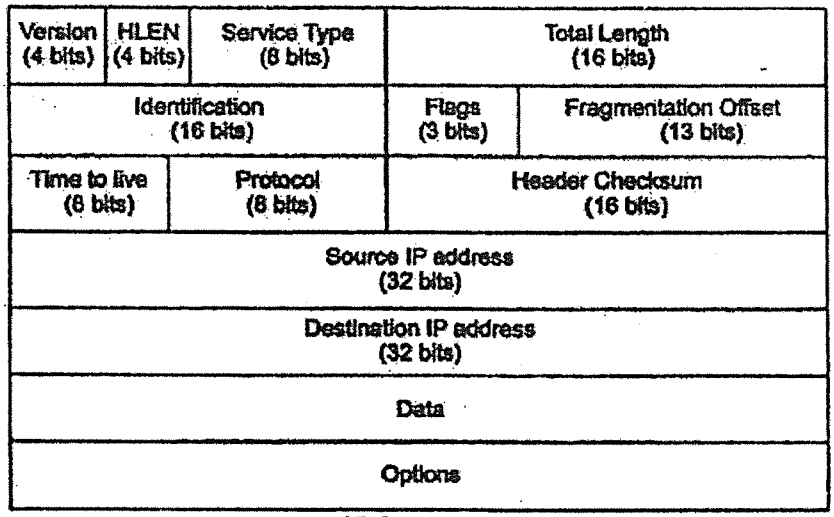
TCP segment format

Explanation:

Item	Description
Source Port	Identifies the port number of a source application program.
Destination Port	Identifies the port number of a destination application program.
Sequence Number	Specifies the sequence number of the first byte of data in this segment.
Acknowledgment Number	Identifies the position of the highest byte received.
Data Offset	Specifies the offset of data portion of the segment.
Reserved	Reserved for future use.
Code	*Control bits to identify the purpose of the segment*:
	URG: Urgent pointer field is valid.
	ACK: Acknowledgement field is valid.
	PSH: Segment requests a PUSH.
	RST: Resets the connection.
	SYN: Synchronizes the sequence numbers.
	FIN: Sender has reached the end of its byte stream.
Window	Specifies the amount of data the destination is willing to accept.
Checksum	Verifies the integrity of the segment header and data.
Urgent Pointer	Indicates data that is to be delivered as quickly as possible. This pointer specifies the position where urgent data ends.
Options	**End of Option List**
	Indicates the end of the option list. It is used at the final option, not at the end of each option individually. This option needs to be used only if the end of the options would not otherwise coincide with the end of the **TCP** header.
	No Operation
	Indicates boundaries between options. Can be used between other options; for example, to align the beginning of a subsequent option on a word boundary. There is no guarantee that senders will use this option, so receivers must be prepared to process options even if they do not begin on a word boundary.
	Maximum Segment Size
	Indicates the maximum segment size **TCP** can receive. This is only sent in the initial connection request.

IP Datagram Format:

The following figure describes the *IP Datagram*:



IP datagram

Explanation:

1. Version - currently has the value 4
2. Header length - the number of 32-bit words in the header
   i. because this is 4 bits, the max header length is 15 words (i.e. 60 bytes)
   ii. the header is at least 20 bytes, but options may make it bigger
3. Type of Service - contains a 3-bit precedence field (that is ignored today), 4 service bits, and 1 unused bit.
   i. The four service bits can be:
   ii. 1000 - minimize delay
   iii. 0100 - maximize throughput
   iv. 0010 - maximize reliability
   v. 0001 - minimize monetary cost
   ii. This is a "hint" of what characteristics of the physical layer to use
   iii. The Type of Service is not supported in most implementations. However, some implementations have extra fields in the routing table to indicate delay, throughput, reliability, and monitory cost.
4. Total Length - specified in bytes.
   i. we know where the data starts by the header length
   ii. we know the size of the data by computing "total length - header length"

5. Identification - uniquely identifies the datagram. Usually incremented by 1 each time a datagram is sent.
6. *Flags and Fragmentation Offset - used for fragmentation (described below)*
7. Time to Live - Upper limit of routers
    i. Usually set to 32 or 64.
    ii. decremented by each router that processes the datagram,
    iii. Router (A **router** is a networking device that forwards data from one network to another) discards the datagram when TTL (***Time to live*** **or** ***hop*** limit is a mechanism that limits the lifespan or lifetime of data in a computer or **network)** reaches 0.
8. Protocol - Tells IP where to send the datagram up to.
    i. 6 means TCP (Transmission Control Protocol)
    ii. 17 means UDP (The User Datagram Protocol (UDP) is one of the core members of the Internet Protocol Suite, the set of network protocols used for the Internet)
9. Header checksum - Only covers the header, not the data.
10. Source IP address -- the sender
11. Destination IP address -- the final destination
12. Options -- optional data.

Firewalls:

"A **firewall** is a network security system, either hardware- or software-based, that controls incoming and outgoing network traffic based on a set of rules."

A firewall may be designed to operate as a filter at the level of IP packets, or may operate at a higher protocol layer.

Firewall Design Principles:

Information systems in corporations, government agencies, and other organizations have undergone a stable development:

1. Centralized data processing system, with a central mainframe supporting a number of directly connected terminals
2. Local area networks (LANs) interconnecting PCs and terminals to each other and the mainframe
3. Premises network, consisting of a number of LANs, interconnecting PCs, servers, and perhaps a mainframe or two.

4. Enterprise-wide network, consisting of multiple, geographically distributed premises networks interconnected by a private wide area network (WAN)
5. Internet connectivity, in which the various premises networks all hook into the Internet and may or may not also be connected by a private WAN

Firewall Characteristics:

[BELL94b] lists the following design goals for a firewall:

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

[SMIT97] lists four general techniques that firewalls use to control access and implement the site's security policy. Originally, firewalls focused primarily on service control, but they have since developed to provide all four:

1. **Service control:** Determines the types of Internet services that can be accessed, inbound or outbound. The firewall may filter traffic on the basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as a Web or mail service.
2. **Direction control:** Determines the direction in which particular service requests may be initiated and allowed to flow through the firewall.
3. **User control:** Controls access to a service according to which user is attempting to access it. This feature is typically applied to users inside the firewall perimeter (local users). It may also be applied to incoming traffic from external users; the latter requires some form of secure authentication technology, such as is provided in IPSec.
4. **Behavior control:** Controls how particular services are used. For example, the firewall may filter e-mail to eliminate spam, or it may enable external access to only a portion of the information on a local Web server.

The following capabilities are within the scope of a firewall:

1. A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks. The use of a single choke point simplifies security management because security capabilities are consolidated on a single system or set of systems.
2. A firewall provides a location for monitoring security-related events. Audits and alarms can be implemented on the firewall system.
3. A firewall is a convenient platform for several Internet functions that are not security related. These include a network address translator, which maps local addresses to Internet addresses, and a network management function that audits or logs Internet usage.
4. A firewall can serve as the platform for IPSec. Using the tunnel mode capability, the firewall can be used to implement virtual private networks.

Limitations of Firewalls:

Firewalls have their limitations, including the following:

1. The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.
2. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.
3. The firewall cannot protect against the transfer of virus-infected programs or files. Because of the variety of operating systems and applications supported inside the perimeter, it would be impractical and perhaps impossible for the firewall to scan all incoming files, e-mail, and messages for viruses.

Types of Firewalls:

The following Figure illustrates the three common types of firewalls: packet filters, application-level gateways, and circuit-level gateways.

1. Packet-Filtering Router:

A packet-filtering router applies a set of rules to each incoming and outgoing IP packet and then forwards or discards the packet. The router is typically configured to filter packets

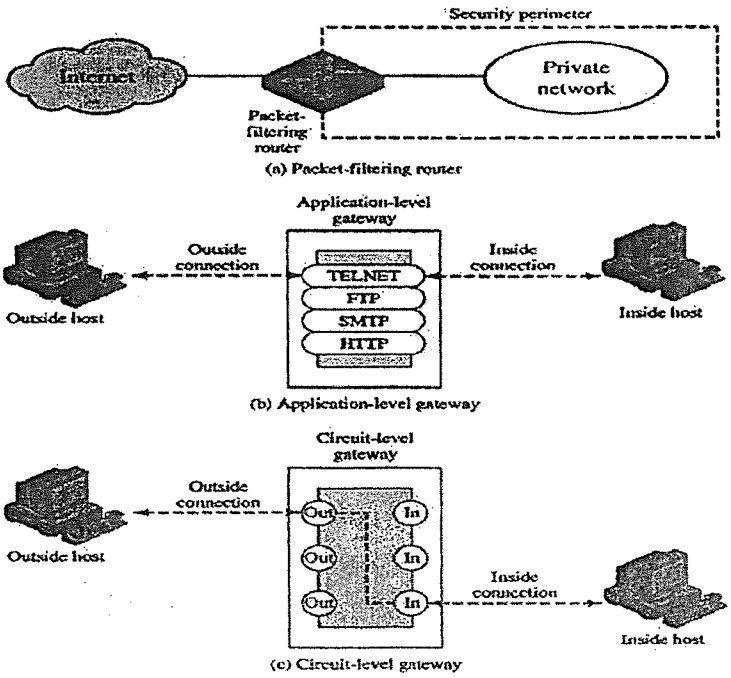
going in both directions (from and to the internal network). Filtering rules are based on information contained in a network packet:

a. Source IP address: The IP address of the system that originated the IP packet (e.g., 192.178.1.1)
b. Destination IP address: The IP address of the system the IP packet is trying to reach (e.g., 192.168.1.2)
c. Source and destination transport-level address: The transport level (e.g., TCP or UDP) port number, which defines applications such as SNMP or TELNET
d. IP protocol field: Defines the transport protocol
e. Interface: For a router with three or more ports, which interface of the router the packet came from or which interface of the router the packet is destined for

The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken. Two default policies are possible:

i. Default = discard: That which is not expressly permitted is prohibited.
ii. Default = forward: That which is not expressly prohibited is permitted.

The default discard policy is more conservative. Initially, everything is blocked, and services must be added on a case-by-case basis. This policy is more visible to users, who are more likely to see the firewall as a hindrance (obstruction).



2. Application-Level Gateway:

An application-level gateway, also called a proxy server, acts as a relay of application-level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.

Application-level gateways tend to be more secure than packet filters. A prime disadvantage of this type of gateway is the additional processing overhead on each connection.

3. Circuit-Level Gateway:

i. This can be a stand-alone system or it can be a specialized function performed by an application-level gateway for certain applications.
ii. relays two TCP connections
iii. requires security by limiting which such connections are allowed
iv. once created usually relays traffic without examining contents
v. typically used when trust internal users by allowing general outbound connections
vi. SOCKS package (Example of this gateway) commonly used for this

Bastion Host:

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

i. highly secure host system
ii. potentially exposed to "hostile" elements
iii. hence is secured to withstand this
iv. may support 2 or more net connections
v. may be trusted to enforce trusted separation between network connections
vi. runs circuit / application level gateways
vii. or provides externally accessible services

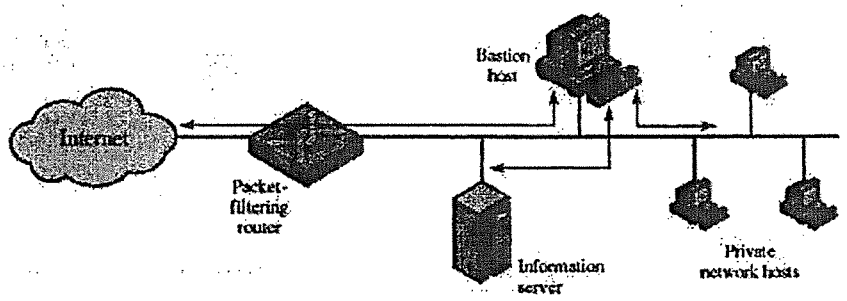
Firewall Configurations:

The following Figure illustrates three common firewall configurations.

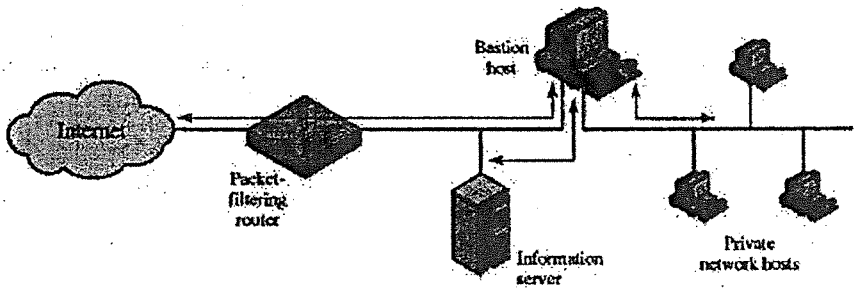
1. Single-homed bastion configuration (Figure: a), the firewall consists of two systems: a packet-filtering router and a bastion host. Typically, the router is configured so that

i. For traffic from the Internet, only IP packets destined for the bastion host are allowed in.

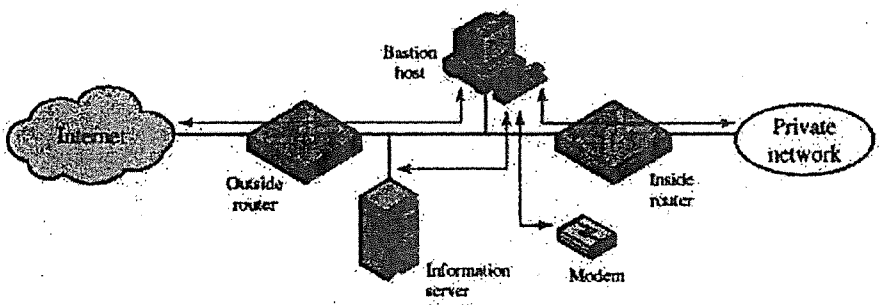
ii. For traffic from the internal network, only IP packets from the bastion host are allowed out.



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



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



(c) Screened-subnet firewall system

2. In the single-homed configuration, if the packet-filtering router is completely compromised, traffic could flow directly through the router between the Internet and other hosts on the private network. The screened host firewall, dual-homed bastion configuration physically prevents such a security break (Figure: b).The advantage of dual layers of security that were present in the previous configuration are present here as well. Again, an information server or other hosts can be allowed direct communication with the router if this is in accord with the security policy.

3. The screened subnet firewall configuration of Figure: c is the most secure. In this configuration, two packet-filtering routers are used, one between the bastion host and the

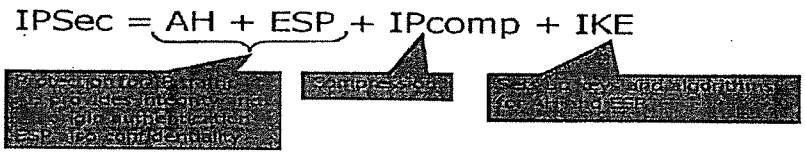
Internet and one between the bastion host and the internal network. This configuration offers several advantages:

i. There are now three levels of defense to spoil intruders.
ii. The outside router advertises only the existence of the screened subnet to the Internet; therefore, the internal network is invisible to the Internet.
iii. Similarly, the inside router advertises only the existence of the screened subnet to the internal network; therefore, the systems on the inside network cannot construct direct routes to the Internet.

IP Security:

"*IP Security (IPSec)* is a collection of protocols designed by the Internet Engineering Task Force (IETF) to provide security for a packet at the network level."

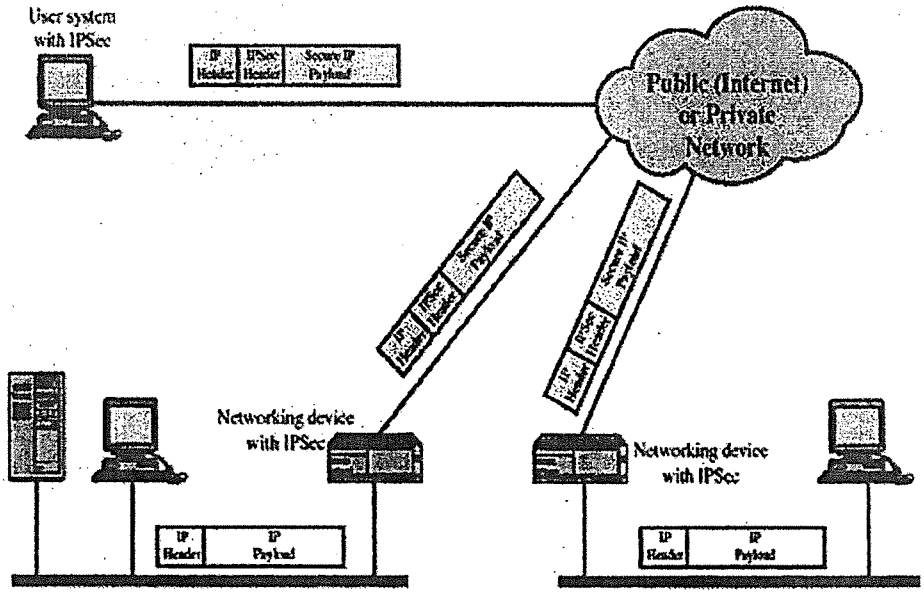
The network layer in the Internet is often referred to as the Internet protocol or IP layer. IPSec helps create authentication and confidential packets for the IP layer.



IP Security Issues:

1. Eavesdropping
2. Modification of packets in transit
3. Identity spoofing (forged source IP addresses)
4. Denial of service
5. Many solutions are application-specific: TLS (Transport Layer Security) for Web, S/MIME (Secure / Multipurpose Internet Mail Extension) for email, SSH (*Se*cure *Sh*ell) for remote login
6. IPSec aims to provide a framework of open standards for secure communications over IP
7. Protect every protocol running on top of IPv4 and IPv6

An IP Security Scenario:



IPSec Document Overview:

The IP Security is very complex. We start at the documents that define IP Security.

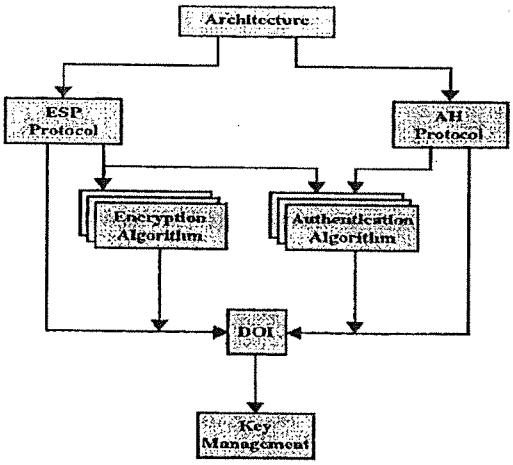
1. RFC 2045 (Request for Comments) – An overview of security architecture
2. RFC 2046 – Description packet authentication
3. RFC 2047 – A specific authentication
4. RFC 2048 – Description of packet encryption
5. RFC 2049 – A specific encryption mechanism

All these documents are published in August, 1995. They define how to provide security at the internet level.

All the IP protocols must support these documents. In IPv6, all these documents are followed, whereas in IPv4 only some are followed.

In both these cases, the security features are implemented as *extension headers* that follow the IP header. The extension header for authentication is known as "*Authentication Header (AH) protocol*".

The extension header for encryption is known as ESP protocol (Encapsulating Security Payload). These documents are divided into 7 groups as shown in the following diagram.



The above architecture covers the general concepts, security requirements, definitions, mechanisms that defines IP Security.

1. *ESP (Encapsulating Security Payload) protocol* covers the packet format & general issues relating to use ESP for packet encryption & optionally packet authentication.
2. *AH (Authentication Header) protocol* covers packet format & general issues related to use AH for packet authentication.
3. *Encryption Algorithm* is a set of documents describes how various encryption algorithms are used in ESP.
4. *Authentication Algorithm* is a set of documents describes how various authentication algorithms are used in AH protocol, optionally in ESP.
5. DOI (Domain of Interpretation) - Contains values needed for other documents to relate to each other. It consists "approved encryption and authentication algorithms and operational parameters such as key lifetime".
6. *Key Management* describes various key management schemes.

Applications of IP Security:

IP Security provides capability of secured communication over the LAN, communication across some private network stations, documentation from private LAN to WAN and across the internet. The following are the examples:

1. Using IP Security, the company can build a secured private network with that he can communicate with another WAN or Internet or LAN.
2. An end user can communicate with Internet.

The main feature of IP Security is to support various applications that can encrypt & authenticate at the IP level.

Benefits of IP Security:

1. When implemented in a firewall or router, it provides security to all traffic crossing the firewall/router perimeter
2. Because IP Security is below the transport layer, it is transparent to applications.
3. IP Security can be transparent to end users, thereby reducing the need to train users on security mechanisms, issue keys on a per-user basis, or revoke keys when users leave the organization.
4. IP Security can provide security to individual users if required.
5. Additional capabilities when applied to a router:
6. Only an authorized router can advertise the presence of a new router.
7. Only an authorized router can advertise the request that establishes a new relationship between two routers.
8. A redirect message must come from the router to which the initial packet was sent.
9. A routing update cannot be forged.

Suppose user B receives an IP Security protected packet from one or more sources. Then B has to know the cryptographic key and algorithm to process that packet. This is done by inserting IP Security header in the packet. IP security packet may be AH (Authentication Header) or ESP (Encapsulation Security Payload). The following concepts are included in IP Security.

1. Security Association (SA), SA Database
2. Security Policy Database
3. AH and ESP
4. Transport and Tunnel Mode

1. SA: Security Association (SA) is a one-way relationship between a sender and a receiver that provides security services to the traffic carried on it.

a. Security association (SA) defines
b. Protocol used (AH, ESP)
c. Mode (transport, tunnel)
d. Encryption or hashing algorithm to be used
e. Negotiated keys and key lifetimes
f. Lifetime of this SA
g. ... plus other info

A system implementing IP security must keep a SA Database. When transmitting to destination 'X', the sender looks up for 'X' in the Table or SA Database. It consists

information about how to transmit to 'X' i.e., it will provide SPI (Security Parameter Index), a key of 'X' and an algorithm, a sequence number etc., Now the sender will use the key, algorithm and send the packet to the 'X'. After receiving, the 'X' will again contact the SA database to get key algorithm and use that information to get the plain data.

2. Security Policy Database (SPD): It contains entries which define the SA for all current traffic. Just like in firewalls, we can configure or program , so that, only some packets are received and some packets are sending out. This type of security policies are defined in the Security Policy Database. All the traffic whether incoming / outgoing must satisfy Security Policy in the Database.

3. AH and ESP: The AH (Authentication Header) is defined in RFC 2402 (*R*equest *F*or Comment) and ESP (Encapsulating Security Payload) defined in RFC 2406. These are two types of IP security Headers.

AH provides Integrity and Authentication. ESP provides Encryption and Integrity. The integrity is provided by both AH and ESP. But there is a slight difference between Integrity Protection by these two. AH provides Integrity for some fields inside IP Header. Both provide Integrity to the Header.

The elements of AH and ESP is described as the following:

AH:

# octets	
1	next header
1	payload length
2	unused
4	SPI (Security Parameter Index)
4	sequence number
variable	authentication data

a. *next header or version* – It indicates IP version which is 4 or 6 (IPv4 or IPv6) and it occupies 1 octet.
b. *payload length* – It denotes size of AH header and it occupies one octet.
c. *SPI & sequence number* – We can get these from SA database.
d. *authentication data* – It is the data for which we are going to provide Integration. It may have any length.

ESP:

# octets	
4	SPI (Security Parameter Index)
4	sequence number
variable	IV (initialization vector)
variable	data
variable	padding
1	padding length (in units of octets)
1	next header / protocol type
variable	authentication data

a. *SPI & sequence number* – Same as for AH. We can get these from SA database.

b. *IV* – Initialization Vector which is used in the cryptographic encryption algorithm such as CBC.

c. *data* – It is protected data, probably encrypted.

d. *padding* – Padding is used for several reasons: to make the data multiple of block size for cryptographic algorithm that require it.

e. *padding length* – Number of octets of padding.

f. *next header* – Same as in AH.

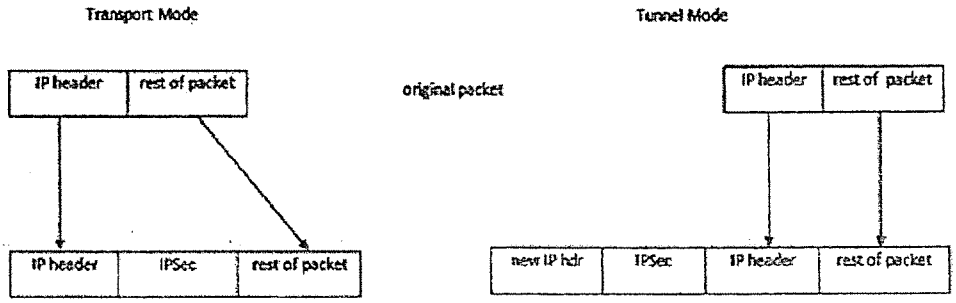
g. *authentication data* – Same as AH. It is zero length if ESP is providing encryption.

4. Transport and Tunnel Mode: There are two types of modes in IP Security. They are

a. Transport Mode – It provides protection for upper layer protocols in the IP packet. In this mode, the IP Security information is placed between the IP Header and rest of the data.

b. Tunnel Mode – It provides protection to the entire IP packet. In this mode, IP Security information is appended to the original packet with a new header and IP Security information.

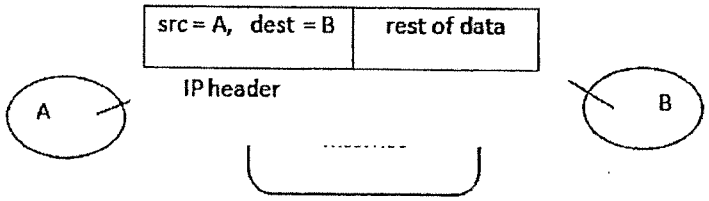
This is explained in the following figure:



Transport mode is used when communicating to end-to-end. Tunnel mode is used when communicating between firewall-to-firewall and end-to-end.

Suppose two *firewalls (F1, F2)* are established in two organizations. User A is in one organization wants to communicating with User B in another organization.

Here organization packet looks like as the following:



The organization packet from user 'A' must pass through F1-Internet-F2 to reach 'B'. The firewall F1will attaches the secured information to the organization packet in the following way using Tunnel mode.

IP: src = F1, dest = F2	ESP	IP: src = A, dest = B	rest of data

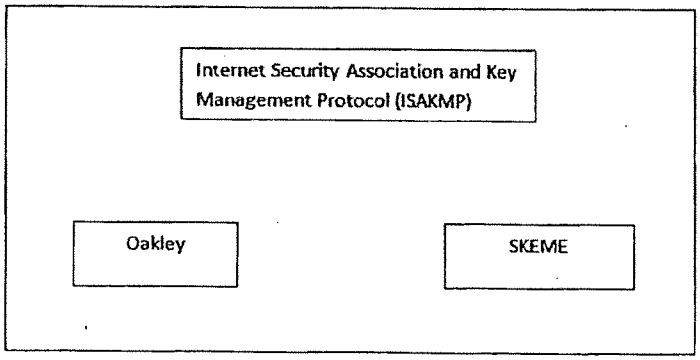
In this way, Transport and Tunnel modes are used while communicating between two ends.

IPsec:IKE:

The *Internet Key Exchange (IKE)* is a *protocol* designed to create both inbound and outbound Security Associations (SAs) i.e., IKE creates SAs for IP Security.

Components of IKE:

The following diagram shows the components of IKE.



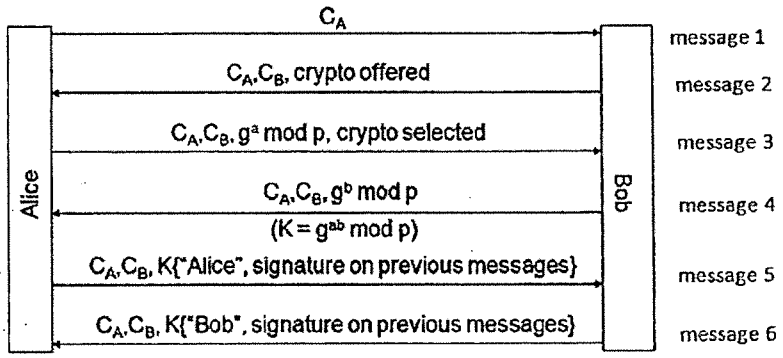
1. The *Oakley* protocol was developed by Hilarie Orman. It is a key creation protocol based on the Diffie-Hellman key-exchange method.

2. The *SKEME* (Secure Key Exchange Mechanism), designed by Hugo Krawcyzk, is another protocol for key exchange. It uses public-key encryption for entity authentication in a key-exchange protocol.
3. The *ISAKMP* (Internet Security Association and Key Management) protocol designed by the National Security Agency (NSA) that actually implements the exchanges defined IKE. It defines several packets, protocols and parameters that allow the IKE exchanges to take place in standardized, formatted messages to create SAs.

PHOTURIS:

Photuris was one of the two main candidates for this piece of IPsec (the other being SKIP). Photuris was basically a signed Diffie-Hellman (DH) exchange, with identity hiding by first doing anonymous Diffie-Hellman and using an initial stateless cookie (cookie – *A text that holds some information about the receiver and must be returned to the sender untouched*).

Alice transmits C_A ,which Photuris calls a cookie, but it is not for the same purpose as Bob's stateless cookie C_B. The concept is explained in the following diagram.



C_A: Alice's cookie; for connection ID
C_B: Bob's cookie

The features of Photuris are

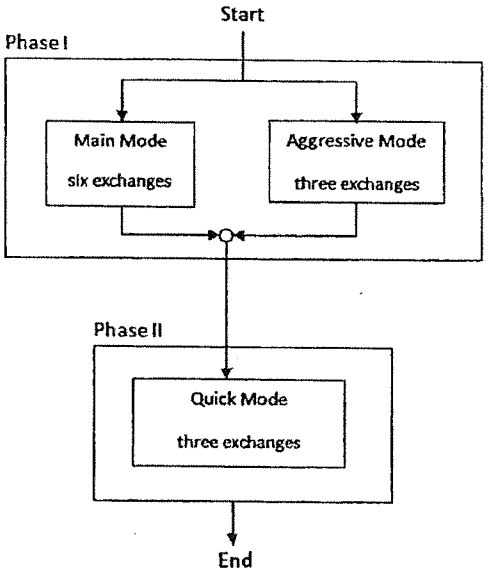
1. Protection by cookies.
2. Authentication & integrity protection of the messages by a combined signature at the last rounds.
3. Identity hiding from passive attackers.

SKIP:

SKIP (Simple Key-Management for Internet Protocols) has some interesting ideas and was at one point widely deployed. SKIP uses long term Diffie-Hellman public keys (e.g., g^a mod p).

IKE Phases:

IKE is divided into two phases. They are Phase I and Phase II. Phase I creates SAs for Phase II. Phase II creates SAs for a data exchange protocol such as IPsec.



Phase 1:

In phase 1, it can do the following things:

1. Does authenticated DH, establishes session key & "ISAKMP SA"
2. There are two possible modes in Phase 1: Main & Aggressive
3. The following two keys are derived from the session key:
    a. SKEYID_e: to encrypt Phase 2 messages
    b. SKEYID_a: to authenticate Phase 2 messages

Phase 2:

In Phase 2, it can do the following things:

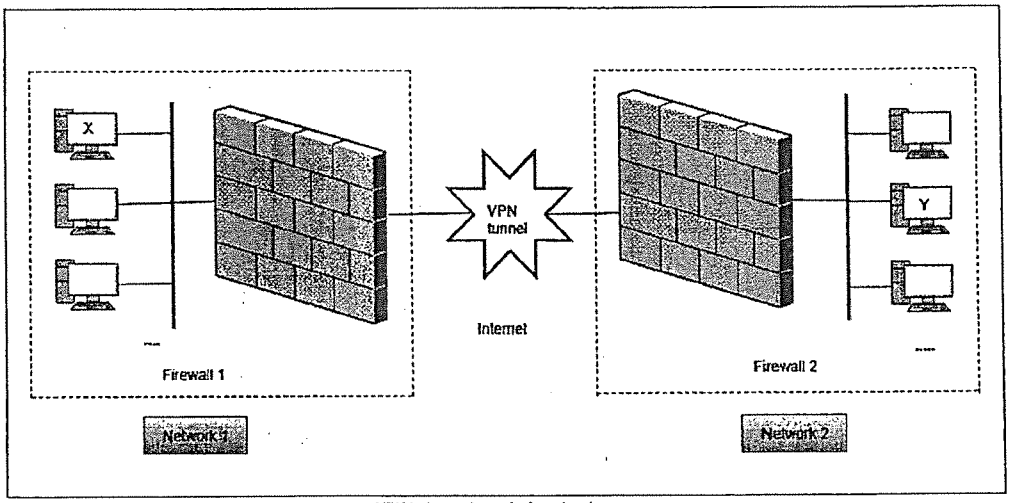
a. IPsec SA & session key established; messages encrypted & authenticated with Phase 1 keys.
b. Additional DH exchange is optional (for PFS).

Virtual Private Networks (VPNs):

A virtual private network (VPN) is a technology that creates a safe and encrypted connection over a less secure network, such as the internet. VPN technology was developed as a way to allow remote users and branch offices to securely access corporate applications and other resources. To ensure safety, data travels through secure tunnels and VPN users must use authentication methods including passwords, tokens and other unique identification methods to gain access to the VPN.

VPN Architecture:

The idea of VPN is actually quite simple to understand. Suppose an organization has two networks, Network 1 and network 2, which are physically aprt from each other and we want to connect them using the VPN approach. In such a case, we setup two firewalls, Firewall 1 and Firewall 2. The encryption and decryption are performed by the firewalls. The architectural overview is shown in the following figure:

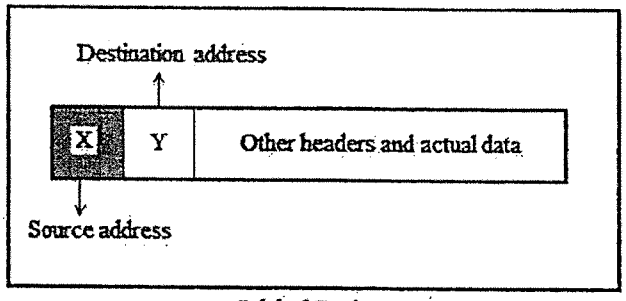


VPN between two private networks

We have shown two networks, Network 1 and Network 2. Network 1 connects to the Internet via a firewall named Firewall 1. Similarly, Network 2 connects to the Internet with its own firewall, Firewall 2. The two firewalls are virtually connected to each other via the Internet. We have shown this with the help of a VPN tunnel between the two firewalls.

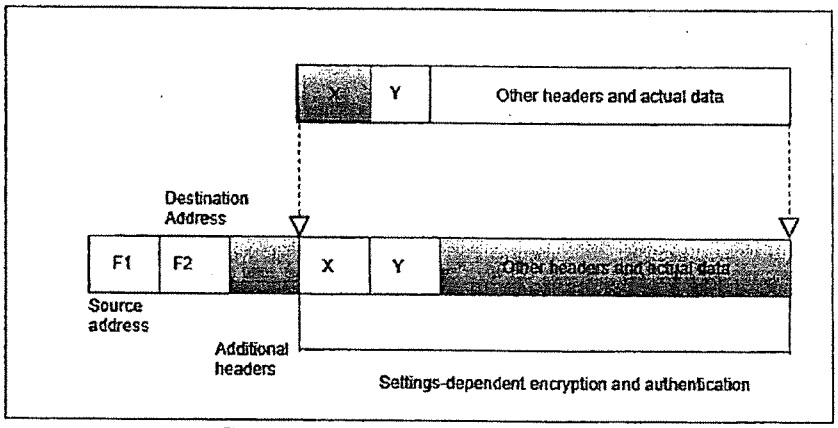
With this configuration, let us understand how VPN protects the traffic passing between any two hosts on the two different networks. For this, let us assume that host X on Network 1 wants to send a data packet to host Y on Network2. This transmission would work as follows:

1. Host X creates the packet, insects its own IP address as the source address and the IP address of host Y as the destination address. This is shown in the following figure. It sends the packet using the appropriate mechanism.



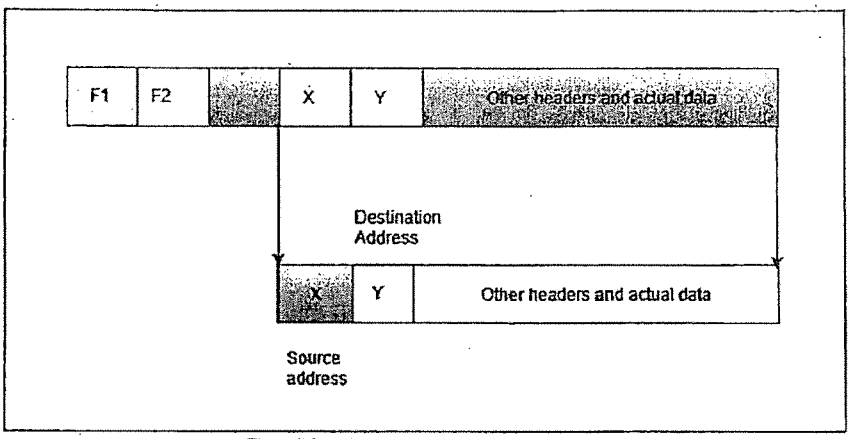
Original Packet

2. The packet reaches Firewall 1. Now Firewall 1 adds new headers to the packet. In these new headers, it changes the source IP address of the packet from that of host X to its own the IP address (i.e. the IP address of Firewall 1, say F_1). It also changes the destination IP address of the packet from that of host Y to the IP address of Firewall 2, say F_2). This is show in the following figure. It also performs the packet encryption and authentication, depending on the settings and sends the modified packet over the Internet.



Firewall 1 changes the original packet contents

3. The packet reaches Firewall 2 over the Internet, via one or more routers, as usual. **Firewall 2** discards the outer header and performs the appropriate decryption and other cryptographic functions as necessary. This yields the original packet, as was created by host X in Step 1. This is shown in the following figure. It then takes a look at the plain text contents of the packet and realizes the packet is meant for host Y (because the destination address inside the packet specific host Y). Therefore, it delivers the packet to host Y.



Firewall 2 retrieves the original packet contents

VPN Protocols: There are mainly three types of protocols. They are

1. **Point – to – Point Tunneling Protocol (PPTP):** PPTP or Point-to-Point Tunneling Protocol creates a tunnel and encapsulates the data packet. It uses a Point-to-Point Protocol (PPP) to encrypt the data between the connections. PPTP is one of the most widely used VPN protocol and has been in use since the time of Windows 95. Apart from Windows, PPTP is also supported on Mac and Linux.
2. **Layer 2 Tunneling Protocol (L2TP):** L2TP or Layer 2 Tunneling Protocol is a tunneling protocol that is usually combined with another VPN security protocol like IPSec to create a highly secure VPN connection. L2TP creates a tunnel between two L2TP connection points and IPSec protocol encrypts the data and handles secure communication between the tunnels.
3. **Internet Protocol Security or IPSec:** Internet Protocol Security or IPSec is used to secure Internet communication across an IP network. IPSec secures Internet Protocol communication by authenticating the session and encrypts each data packet during the connection. IPSec operates in two modes, Transport mode and Tunneling mode, to protect data transfer between two different networks. The transport mode encrypts the message in the data packet and the tunneling mode encrypts the entire data packet. IPSec can also be used with other security protocols to enhance the security system.

****