UNIT-I

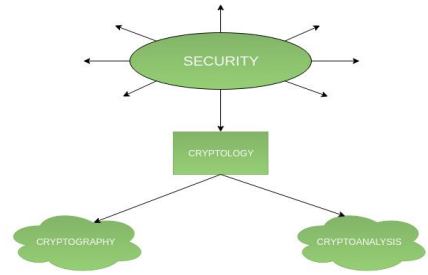
## ➢ Cryptography

Cryptography is the study of secure communications techniques that allow only the sender and intended recipient of a message to view its contents. The term is derived from the Greek word *kryptos*, which means hidden. It is closely associated to encryption, which is the act of scrambling ordinary text into what's known as ciphertext and then back again upon arrival. In addition, cryptography also covers the obfuscation of information in images using techniques such as microdots or merging. Ancient Egyptians were known to use these methods in complex hieroglyphics, and Roman Emperor Julius Caesar is credited with using one of the first modern ciphers.

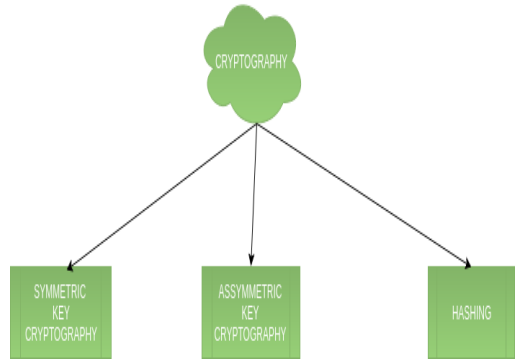
When transmitting electronic data, the most common use of cryptography is to encrypt and decrypt email and other plain-text messages. The simplest method uses the symmetric or "secret key" system. Here, data is encrypted using a secret key, and then both the encoded message and secret key are sent to the recipient for decryption. The problem? If the message is intercepted, a third party has everything they need to decrypt and read the message. To address this issue, cryptologists devised the asymmetric or "public key" system. In this case, every user has two keys: one public and one private. Senders request the public key of their intended recipient, encrypt the message and send it along. When the message arrives, only the recipient's private key will decode it — meaning theft is of no use without the corresponding private key.

## ➢ Introduction to Crypto-terminologies

Cryptography is an important aspect when we deal with network security. 'Crypto' means secret or hidden. Cryptography is the science of secret writing with the intention of keeping the data secret. Cryptanalysis, on the other hand, is the science or sometimes the art of breaking cryptosystems. These both terms are a subset of what is called as Cryptology.




Classification –
The flowchart depicts that cryptology is only one of the factors involved in securing networks. Cryptology refers to study of codes, which involves both writing (cryptography) and solving (cryptanalysis) them. Below is a classification of the crypto-terminologies and their various types.



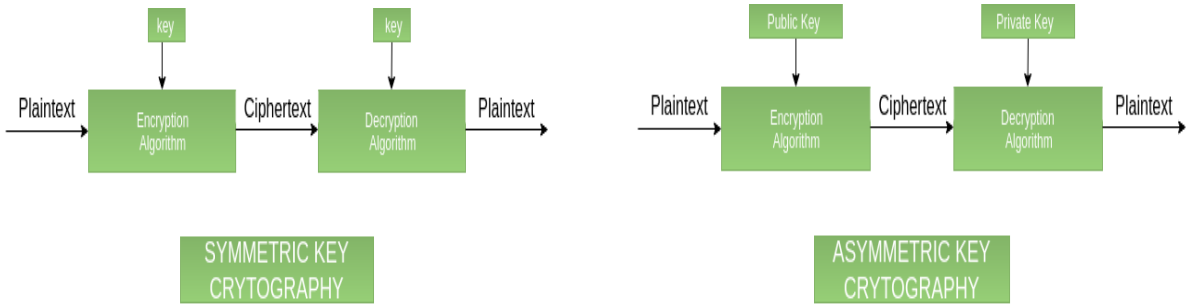

**1.** Cryptography –
Cryptography is classified into symmetric cryptography, asymmetric cryptography and hashing. Below are the description of these types.

1. Symmetric key cryptography

It involves usage of one secret key along with encryption and decryption algorithms which help in securing the contents of the message. The strength of symmetric key cryptography depends upon the number of key bits. It is relatively faster than asymmetric key cryptography. There arises a key distribution problem as the key has to be transferred from the sender to receiver through a secure channel.

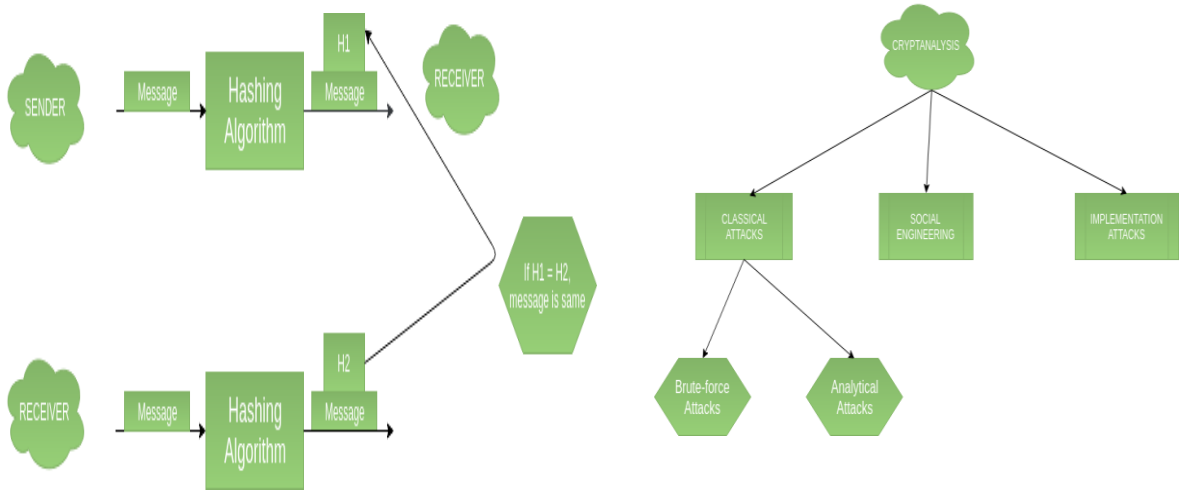
2. Asymmetric key cryptography

It is also known as public key cryptography because it involves usage of a public key along with secret key. It solves the problem of key distribution as both parties uses different keys for encryption/decryption. It is not feasible to use for decrypting bulk messages as it is very slow compared to symmetric key cryptography.




3.Hashing –

It involves taking the plain-text and converting it to a hash value of fixed size by a hash function. This process ensures integrity of the message as the hash value on both, sender\'s and receiver\'s side should match if the message is unaltered.




2. Cryptanalysis –

1.Classical attacks

It can be divided into a)Mathematical analysis and b) Brute-force attacks. Brute-force attacks run the encryption algorithm for all possible cases of the keys until a match is found. Encryption algorithm is treated as a black box. Analytical attacks are those attacks which focus on breaking the cryptosystem by analysing the internal structure of the encryption algorithm.

2.Social Engineering attack

It is something which is dependent on the human factor. Tricking someone to reveal their passwords to the attacker or allowing access to the restricted area comes under this attack. People should be cautious when revealing their passwords to any third party which is not trusted.

3.Implementation attacks

Implementation attacks such as side-channel analysis can be used to obtain a secret key. They are relevant in cases where the attacker can obtain physical access to the cryptosystem.

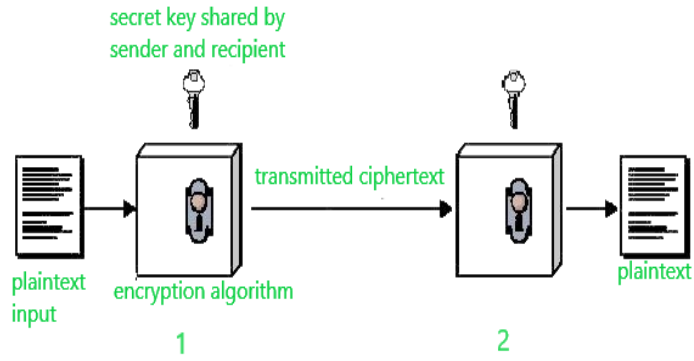
## ➢Conventional Encryption Model

Conventional encryption is a cryptographic system that uses the same key used by the sender to encrypt the message and by the receiver to decrypt the message. It was the only type of encryption in use prior to the development of public-key encryption.

It is still much preferred of the two types of encryption systems due to its simplicity. It is a relatively fast process since it uses a single key for both encryption and decryption In this encryption model, the sender encrypts plaintext using the receiver's secret key, which can be later used by the receiver to decrypt the ciphertext. Below is a figure that illustrates this concept.

Suppose A wants to send a message to B, that message is called plaintext. Now, to avoid hackers reading plaintext, the plaintext is encrypted using an algorithm and a secret key (at 1). This encrypted plaintext is called ciphertext. Using the same secret key and encryption algorithm run in reverse(at 2), B can get plaintext of A, and thus the message is read and security is maintained.

The idea that uses in this technique is very old and that's why this model is called conventional encryption.




**Conventional encryption has mainly 5 ingredients :**

**Plain text –**

It is the original data that is given to the algorithm as an input.

**Encryption algorithm –**

This encryption algorithm performs various transformations on plain text to convert it into ciphertext.

**Secret key –**

The secret key is also an input to the algorithm. The encryption algorithm will produce different outputs based on the keys used at that time.

**Ciphertext –**

It contains encrypted information because it contains a form of original plaintext that is unreadable by a human or computer without proper cipher to decrypt it. It is output from the algorithm.

**Decryption algorithm –**

This is used to run encryption algorithms in reverse. Ciphertext and Secret key is input here and it produces plain text as output.

**Requirements for secure use of conventional encryption:**

1.We need a strong encryption algorithm.

2. The sender and Receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure.

**Advantages of Conventional Encryption :**

1.Simple –

This type of encryption is easy to carry out.

2.Uses fewer computer resources –

Conventional encryption does not require a lot of computer resources when compared to public-key encryption.

3.Fast –

Conventional encryption is much faster than asymmetric key encryption.

**Disadvantages of Conventional Encryption Model:**

1.Origin and authenticity of the message cannot be guaranteed, since both sender and receiver use the same key, messages cannot be verified to have come from a particular user.

2.It isn't much secured when compared to public-key encryption.

3.If the receiver lost the key, he/she cant decrypt the message and thus making the whole process useless.

4.This scheme does not scale well to a large number of users because both the sender and the receiver have to agree on a secret key before transmission.

## ➢Steganography

Steganography is a technique of hiding communication by concealing the secret message into a fake message. The term Steganography has a Greek influence which means "covered writing". The main idea behind the Steganography is to prevent suspicion about the existence of the information.

- Pure Steganography does not require the exchange of a cipher such as a stego-key. It assumes that no other party is aware of the communication.
- Secret key Steganography where the secret (stego) key is exchanged prior to communication. This is most susceptible to interception. Secret Key Steganography takes a cover message and embeds the secret message inside of it by using a secret key (stego-key). Only the parties who know the secret key can reverse the process and read the secret message.
- Public key Steganography where a public key and a private key is used for secure Communication. The sender will use the public key during the encoding process and only the private key, which has a direct mathematical relationship with the public key, can decipher the secret message.

**Types of Steganography**

**Image Steganography**

- The image Steganography is used to hide a secret message inside an image. The most widely used technique to hide secret bit inside the LSB of the cover image. Because this method uses bits of each pixel in the image, it is necessary to use a lossless compression format, otherwise the hidden information will get lost in the transformations of a lossy compression algorithm.
- When using a 24 bit color image, a bit of each of the red, green and blue color components can be used, so a total of 3 bits can be used for each pixel, in this way we can use more secret bit to hide data in it.

**Audio Steganography**

- Audio stenography can conceal the secret message in the audio file with the help of its digital representation. It can be achieved easily as a typical 16-bit file has 216 sound levels, and a few levels difference could not be detectable by the human ear.
- The sender embeds secret data of any type using a key in a digital cover file to produce a stego file, in such a way that an observer cannot detect the existence of the hidden message. In many schemes a method of audio Steganography based on modification of least significant bits (LSB) the audio samples in the temporal domain or transform domain have been proposed.

**Video Steganography**

- Video Steganography brings more possibilities of disguising a large amount of data because it is a combination of image and sound. Therefore, image and audio Steganography techniques can also be employed on the video.
- Video files are generally a collection of images and sounds, so most of the presented techniques on images and - audio can be applied to video files too.
- The great advantage of video is the large amount of data that can be hidden inside and the fact that it is a moving stream of images and sounds.
- The Video Steganography is nothing but a combination of Image Steganography and Audio Steganography.

**Text Steganography:**

- Steganography can be applied to different types of media including text, audio, image and video etc. However, text Steganography is considered to be the most difficult kind of Steganography due to lack of redundancy in text as compared to image or audio but still has smaller memory occupation and simpler communication.
- The method that could be used for text Steganography is data compression. Data compression encodes information in one representation into another representation. The new representation of data is smaller in size.
- One of the possible schemes to achieve data compression is Huffman coding. Huffman coding assigns smaller length code words to more frequently occurring source symbols and longer length code-words to less frequently occurring source symbols.

## ➢ Classical Encryption Techniques: A SYMMETRIC CIPHER MODEL:

Symmetric encryption, also referred to as conventional encryption or single-key encryption, was the only type of encryption in use prior to the development of public key encryption in the1970s.

Some basic terminologies used:

- ciphertext - the coded message
- cipher - algorithm for transforming plaintext to ciphertext
- key - info used in cipher known only to sender/receiver
- encipher (encrypt) - converting plaintext to ciphertext
- decipher (decrypt) - recovering ciphertext from plaintext
- cryptography - study of encryption principles/methods
- cryptanalysis (code breaking) - the study of principles/ methods of deciphering ciphertext without knowing key
- cryptology - the field of both cryptography and cryptanalysis

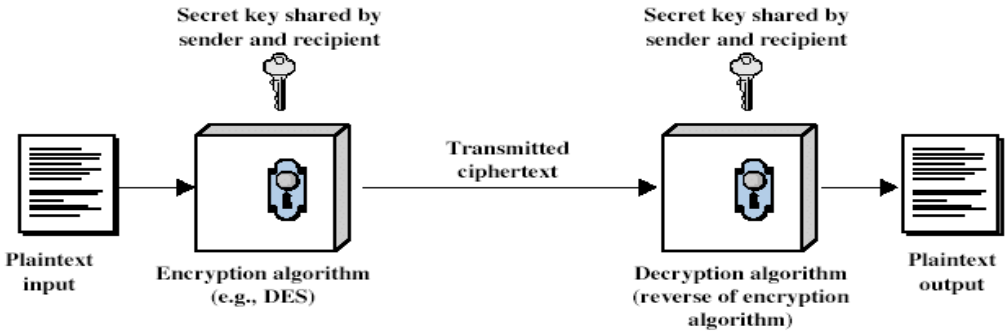



Fig.7 Simplified Model of Symmetric Encryption

**A symmetric encryption scheme has five ingredients**

A symmetric encryption scheme has five ingredients (Fig.7). 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.

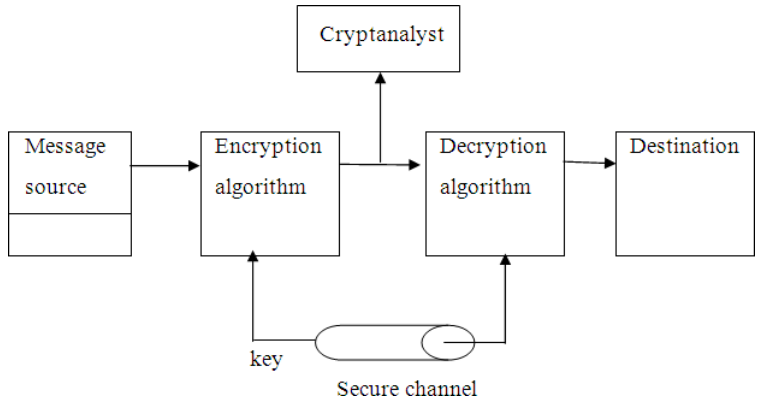
The security depends on several factors. First, the encryption algorithm must be powerful enough that it is impractical to decrypt a message on the basis of cipher text alone. Beyond that, the security depends on the secrecy of the key, not the secrecy of the algorithm.

Two requirements for secure use of symmetric encryption:

- A strong encryption algorithm
- A secret key known only to sender / receiver
- Y = EK(X)
- X = DK(Y)

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

Fig.8. conventional cryptosystem




A source produces a message in plaintext, X = [X1, X2, …, XM] where M are the number of letters in the message. A key of the form K = [K1, K2, …, KJ] 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 = [Y1, Y2, …, YN]. This can be expressed as Y = EK(X)

The intended receiver, in possession of the key, is able to invert the transformation: X = DK(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 anestimate.

**Substitution Encryption Techniques:**

Substitution encryption technique is one type of classic encryption technique, 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 ciphertext bit patterns.

- (i)Caesar cipher (or) shift cipher
- The earliest known use of a substitution cipher and the simplest was by Julius Caesar.
- The Caesar Cipher is a type of shift cipher. Shift Ciphers work by using the modulo operator to encrypt and decrypt messages. The Shift Cipher has a key K, which is an integer from 0 to 25. We will only share this key with people that we want to see our message
- The Caesar cipher involves replacing each letter of the alphabet with the letter standing 3 places further down the alphabet.
- e.g., Plain text: pay more mone Cipher text: SDB PRUH PRQHB
- Note that the alphabet is wrapped around, so that letter following „z" is „a".
- Note that the alphabet is wrapped around, so that the letter following Z isA.
- We can define the transformation by listing all possibilities, as follows: plain: a b c d e f g h i j k l m n o p q r s t u v w x y z cipher: D E F G H I J K L M N O P Q R S T U V W X Y Z A B C
- Let us assign a numerical equivalent to each letter:

| a | b | c | d | e | f | g | h | i | j | k | l | m |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

| n | o | p | q | r | s | t | u | v | w | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |

For Encrypt each plaintext letter p, substitute the cipher text letter c such that C = E(p) = (p+3) mod 26,
a shift may be any amount, so that general Caesar algorithm isC = E (p) = (p+k) mod 26
where k takes on a value in the range 1 to 25.
The decryption algorithm is simply P = D(C) = (C-k) mod 26 (or) to Encrypt a message M. Convert the letter into the number that matches its order in the alphabet starting from 0, and call this number X, (A=0, B=1, C=2, ..., Y=24, Z=25).
Calcúlate: Y = (X + K) mod 26

Convert the number Y into a letter that matches its order in the alphabet starting from 0.Example:

By using the Shift Cipher with key K=19 for our message. We encrypt the message "KHAN", as follows

ENCRYPTION

| | K | H | A | N | |
|---|---|---|---|---|---|
| | 10 | 7 | 0 | 13 | |
| + | 19 | 19 | 19 | 19 | |
| ( | 29 | 26 | 19 | 32 | ) mod 26 |
| | 3 | 0 | 19 | 6 | |
| | D | A | T | G | |

So, after applying the Shift Cipher with key K=19 our message text "KHAN" gaveus cipher text "DATG".

- For every letter in the cipher text C, convert the letter into the number that matches itsorder in the alphabet starting from 0, and call this number Y.
- If it is known that a given ciphertext is a Caesar cipher, then a brute-force cryptanalysis is easily performed: Simply try all the 25 possible keys.

**Monoalphabetic Ciphers:**

With only 25 possible keys, the Caesar cipher is far from secure. A dramatic increase in the key space can be achieved by allowing an arbitrary substitution. Before proceeding, the term permutation can be defined.

A permutation of a finite set of elements S is an ordered sequence of all the elements of S, with each element appearing exactly once.

For example, if S = {a, b, c}, there are six permutations of S:abc, acb, bac, bca, cab, cba

In general, there are n! permutations of a set of n elements, because the first element can be chosen in one of n ways, the second in n - 1 ways, the third in n – 2 ways, and so on.

plain: 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 Caesar cipher: d e f g h i j k l m n o p q r s T u v w x y z a b c

If, instead, the “cipher” line can be any permutation of the 26 alphabetic characters, then there are 26! or greater than 4 * 1026 possible keys.

This is 10 orders of magnitude greater than the key space for DES and would seem to eliminate brute-force techniques for cryptanalysis. Such an approach is referred to as a mono alphabetic substitution cipher, because a single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message.

Monoalphabetic ciphers are easy to break because they reflect the frequency data of the original alphabet.

A countermeasure is to provide multiple substitutes known as homophones, for a single letter. For example, the letter e could be assigned a number of different cipher symbols, such as 16, 74, 35, and 21, with each homophone assigned to a letter in rotation or randomly.

**Playfair Cipher:**

The best-known multiple-letter encryption cipher is the Playfair, which treats diagrams in the plaintext as single units and translates these units into cipher text diagrams

he Playfair algorithm is based on the use of a 5 * 5 matrix of letters constructed using a keyword. Here is an example, solved by Lord Peter Wimsey in Dorothy Sayers‟s Have HisCarcase

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

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

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

Repeating plaintext letters that are in the same pair are separated with a filler letter, such as x, so that balloon would be treated as ba lx lo on.

Two plaintext letters that fall in the same row of the matrix are each replaced by the letter to the right, with the first element of the row circularly following the last. For example, ar is encrypted as RM.

Two plaintext letters that fall in the same column are each replaced by the letter beneath, with the top element of the column circularly following the last. For example, mu is encrypted as CM.

Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM

The Playfair cipher is a great advance over simple monoalphabetic ciphers. For one thing, whereas there are only 26 letters, there are 26 * 26 = 676 digrams, so that identification of individual digrams is more difficult. Furthermore, the relative frequencies of individual letters exhibit a much greater range than that of digrams, making frequency analysis muchmore difficult.

For these reasons, the Playfair cipher was for a long time considered unbreakable. It was used as the standard field system by the British Army in World War I and still enjoyed considerable use by the U.S. Army and other Allied forces during World War II.

**Hill Cipher:**

Another interesting multiletter cipher is the Hill cipher, developed by the mathematician Lester Hill in 1929.

**The Hill Algorithm**

This encryption algorithm takes m successive plaintext letters and substitutes for them m ciphertext letters. The substitution is determined by m linear equations in which each character is assigned a numerical value (a = 0, b = 1,….., z = 25). For m = 3, the system canbe described as

$c_1 = (k_{11}p_1 + k_{21}p_2 + k_{31}p_3) \bmod 26$ $c_2 = (k_{12}p_1 + k_{22}p_2 + k_{32}p_3) \bmod 26$ $c_3 = (k_{13}p_1 + k_{23}p_2 + k_{33}p_3) \bmod 26$

This can be expressed in terms of row vectors and matrices:

$$(c_1 c_2 c_3) = (p_1 p_2 p_3) \begin{pmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{pmatrix} \bmod 26$$

or

$$C = PK \bmod 26$$

Where C and P are row vectors of length 3 representing the plaintext and ciphertext, and K is a 3 x 3 matrix representing the encryption key. Operations are performed mod 26.

**Polyalphabetic ciphers**

A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets. The Vigenère cipher is probably the best-known example of a polyalphabeticcipher.

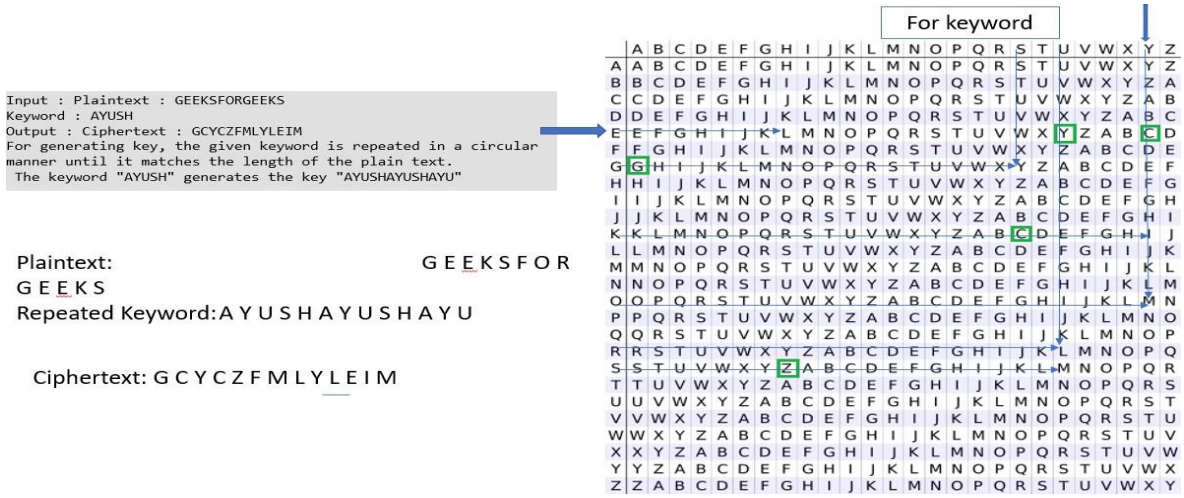
**Difference between monoalphabetic cipher and polyalphabetic cipher:**

A monoalphabetic cipher is a substitution cipher in which the cipher alphabet is fixed through the encryption process.....A polyalphabetic cipher is a substitution cipher in which the cipher alphabet changes during the encryption process.

**Vigenere cipher:**

 Vigenere Cipher is a method of encrypting alphabetic text. It uses a simple form of polyalphabetic substitution. A polyalphabetic cipher is any cipher based on substitution, using multiple substitution alphabets .The encryption of the originaltext is done using the Vigenère square or Vigenère table.

- The table consists of the alphabets written out 26 times in different rows, each alphabet shifted cyclically to the left compared to the previous alphabet, corresponding to the 26 possible Caesar Ciphers.
- At different points in the encryption process, the cipher uses a different alphabet from one of the rows.




- The alphabet used at each point depends on a repeating keyword

The Vigenère cipher can be expressed in the following manner. Assume a sequence of plaintext letters P = $p_0$, $p_1$, $p_2$,.....,$p_{n-1}$ and a key consisting of the sequence of letters K = $k_0$, $k_1$, $k_2$,....., $k_{m-1}$, where typically m <n. The sequence of ciphertext letters C = $C_0$, $C_1$, $C_2$, ……, $C_{n-1}$ is calculated as follows:

C = $C_0$, $C_1$, $C_2$, …..,$C_{n-1}$ = E(K, P) = E[($k_0$, $k_1$, $k_2$, ….., $k_{m-1}$), ($p_0$, $p_1$, $p_2$…., $p_{n-1}$)]

= ($p_0$ + $k_0$) mod 26, ($p_1$ + $k_1$) mod 26, ……, ($p_{m-1}$ + $k_{m-1}$) mod 26, ($p_m$+ $k_0$) mod 26, ($p_{m+1}$ + $k_1$) mod 26, ……. , ($p_{2m-1}$ + km-1) mod 26,…..

Thus, the first letter of the key is added to the first letter of the plaintext, mod 26, the second letters are added, and so on through the first m letters of the plaintext. For the next m letters of the plaintext, the key letters are repeated. This process continues until all of the plaintext sequence is encrypted. A general equation of the encryption process is

$C_i$= ($p_i$+ $k_i$ mod m) mod 26

A general equation for decryption is

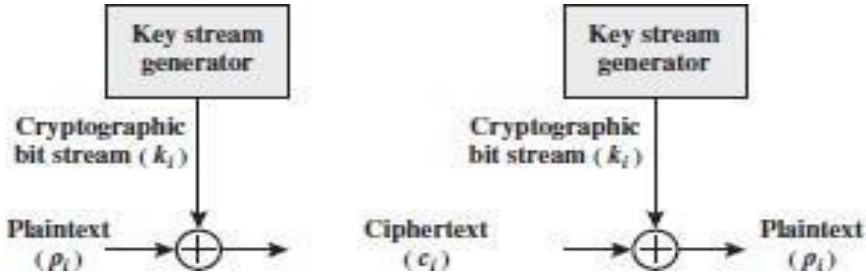
$p_i$= ($C_i$– $k_i$ mod m) mod 26

To encrypt a message, a key is needed that is as long as the message. Usually, the key is a repeating keyword. For example, if the keyword is deceptive, the message "we are discovered save yourself" is encrypted as

Key :deceptivedeceptivedeceptive plaintext :
Wearediscoveredsaveyourselfciphertext: ZICVTWQNGRZGVTWAVZHCQYGLMGJ

The strength of this cipher is that there are multiple ciphertext letters for each plaintext letter, one for each unique letter of the keyword. Thus, the letter frequency information is obscured. However, not all knowledge of the plaintext structure is lost.

Vernam Cipher The ultimate defense against such a cryptanalysis is to choose a keyword that is as long as the plaintext and has no statistical relationship to it. Such a system was introduced by an




AT&T engineer named Gilbert Vernam in 1918.

- The system can be expressed as:

$$ci = pi \oplus ki$$

where

pi= ith binary digit of plaintext
ki= ith binary digit of key
ci= ith binary digit of ciphertext
$\oplus$ s= exclusive-or (XOR) operation

- Thus, the ciphertext is generated by performing the bitwise XOR of the plaintext and the key. Because of the properties of the XOR, decryption simply involves the same bitwise operation

One Time Pad Cipher

It is an unbreakable cryptosystem. It represents the message as a sequence of 0s and 1s. this can be accomplished by writing all numbers in binary, for example, or by using ASCII. The key is a random sequence of 0"s and 1"s of same length as the message. Once a key is used, it is discarded and never used again. The system can be expressed as follows:

Ci = Pi Ki

Ci - ith binary digit of cipher text Pi - ith binary digit of plaintext Ki - ith binary digit of key – exclusive OR opearaiton

Thus, the cipher text is generated by performing the bitwise XOR of the plaintext and the key. Decryption uses the same key. Because of the properties of XOR, decryption simply involves the same bitwise operation:

Pi = Ci Ki

e.g., plaintext = 0 0 1 0 1 0 0 1

Key = 1 0 1 0 1 1 0 0

ciphertext = 1 0 0 0 0 1 0 1

Advantage:

- Encryption method is completely unbreakable for a ciphertext only attack.

Disadvantages

- It requires a very long key which is expensive to produce and expensive to transmit.
- Once a key is used, it is dangerous to reuse it for a second message; any knowledge on the first message would give knowledge of the second.

STEGANOGRAPHY:

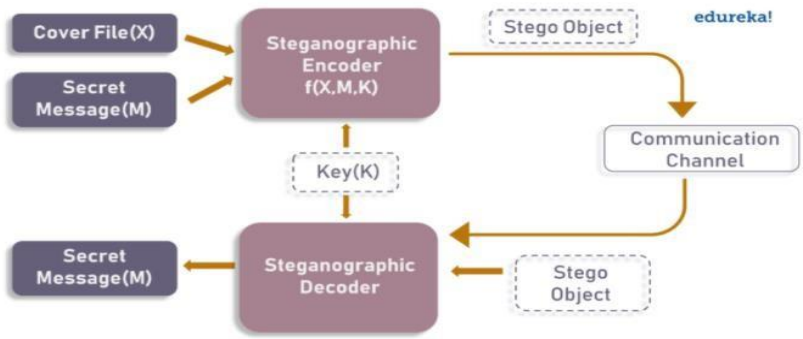
 Steganography is the technique of hiding secret data within an ordinary, non-secret, file or message in order to avoid detection; the secret data is then extracted at its destination.

The use of steganography can be combined with encryption as an extra step for hiding or protecting data.

 It stems from two Greek words, which are steganos, means covered and graphia, means writing

 Examples,

1.Playing an audio track backwards to reveal a secret message
2.Playing a video at a faster frame rate (FPS) to reveal a hidden image
3.Embedding a message in the red, green, or blue channel of an RGB image
4.Hiding information within a file header or metadata
5.Embedding an image or message within a photo through the addition of digital noise




•As the image depicts, both cover file(X) and secret message(M) are fed into steganographic encoder as input.

•Steganographic Encoder function, f(X,M,K) embeds the secret message into a coverfile.

• Resulting Stego Object looks very similar to your cover file, with no visible changes.

• This completes encoding. To retrieve the secret message, Stego Object is fed into Steganographic Decoder.

 Steganography Techniques

 Depending on the nature of the cover object (actual object in which secret data is embedded), steganography can be divided into five types:

1.Text Steganography
2.Image Steganography
3.Video Steganography
4.Audio Steganography
5.Network Steganography

 **Text Steganography**: Text Steganography is hiding information inside the text files. Various techniques used to hide the data in the text are:

•Format Based Method
•Random and Statistical Generation
•Linguistic Method

 **Image Steganography**: Hiding the data by taking the cover object as the image is known as image steganography. There are a lot of ways to hide information inside an image. Common approaches include:

•Least Significant Bit Insertion
•Masking and Filtering
•Redundant Pattern Encoding
•Encrypt and Scatter
•Coding and Cosine Transformation

 **Audio Steganography**: In audio steganography, the secret message is embedded into an audio signal which alters the binary sequence of the corresponding audio file. Different methods of audio steganography include:

•Least Significant Bit Encoding
•Parity Encoding
•Phase Coding
•Spread Spectrum

 **Video Steganography**: In Video Steganography you can hide kind of data into digital video format. Two main classes of Video Steganography include:

•embedding data in uncompressed raw video and compressing it later
•Embedding data directly into the compressed data stream
•Network Steganography (Protocol Steganography): It is the technique of embedding information within network control protocols used in data transmission such TCP, UDP, ICMP etc. For Example, you can hide information in the header of a TCP/IP packet in some fields that are either optional.

Example:

(i) the sequence of first letters of each word of the overall message spells out the real (hidden) message.

(ii) 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:

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. 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.

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**

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

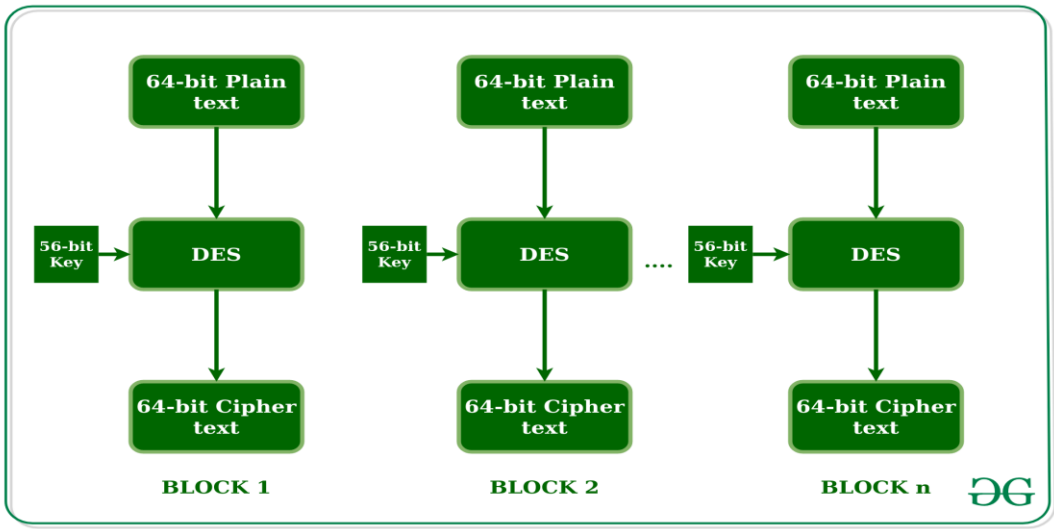
 **TRANSPOSITION TECHNIQUES**:

All the techniques examined so far involve the substitution of a cipher text symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

Rail fence is simplest of such cipher, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.
Plaintext = meet at the school house
To encipher this message with a rail fence of depth 2, we write the message asfollows:
m e a t e c o l o se t t h s H o h u e
The encrypted message is MEATECOLOSETTHSHOHUE
Row Transposition Ciphers-A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of columns then becomes the key of the algorithm.
e.g., plaintext = meet at the school houseKey = 4 3 1 2 5 6 7
PT = m e e t a t th e s c h o o
l h o u s e
CT = ESOTCUEEHMHLAHSTOETO
A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is more complex permutation that is not easily reconstructed

## ➢Data encryption standard (DES)

**Data encryption standard (DES)** has been found vulnerable against very powerful attacks and therefore, the popularity of DES has been found slightly on the decline. DES is a block cipher and encrypts data in blocks of size of 64 bits each, which means 64 bits of plain text goes as the input to DES, which produces 64 bits of ciphertext. The same algorithm and key are used for encryption and decryption, with minor differences. The key length is 56 bits. The basic idea is shown in the figure.



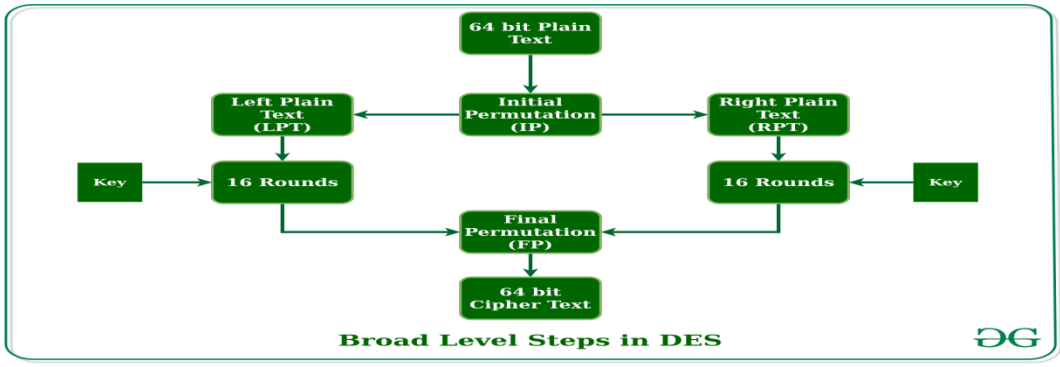

We have mentioned that DES uses a 56-bit key. Actually, the initial key consists of 64 bits. However, before the DES process even starts, every 8th bit of the key is discarded to produce a 56-bit key. That is bit positions 8, 16, 24, 32, 40, 48, 56, and 64 are discarded.

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

**Figure -** discording of every 8th bit of original key

Thus, the discarding of every 8th bit of the key produces a 56-bit key from the original 64-bit key. DES is based on the two fundamental attributes of cryptography: substitution (also called confusion) and transposition (also called diffusion). DES consists of 16 steps, each of which is called a round. Each round performs the steps of substitution and transposition. Let us now discuss the broad-level steps in DES.

1.In the first step, the 64-bit plain text block is handed over to an initial Permutation (IP) function.
2. The initial permutation is performed on plain text.
3. Next, the initial permutation (IP) produces two halves of the permuted block; says Left Plain Text (LPT) and Right Plain Text (RPT).
4. Now each LPT and RPT go through 16 rounds of the encryption process.
5. In the end, LPT and RPT are rejoined and a Final Permutation (FP) is performed on the combined block
6. The result of this process produces 64-bit ciphertext.



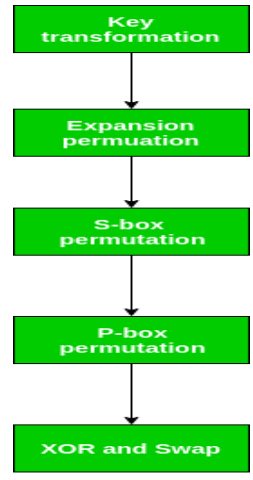

**Initial Permutation (IP)**

As we have noted, the initial permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed, as shown in the figure. For example, it says that the IP replaces the first bit of the original plain text block with the 58th bit of the original plain text, the second bit with the 50th bit of the original plain text block, and so on This is nothing but jugglery of bit positions of the original plain text block. the same rule applies to all the other bit positions shown in the figure.

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 33 | 45 | 37 | 29 | 21 | 13 | 5 | 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 |

**Figure -** Initial permutation table

As we have noted after IP is done, the resulting 64-bit permuted text block is divided into two half blocks. Each half-block consists of 32 bits, and each of the 16 rounds, in turn, consists of the broad level steps outlined in the figure.




**Step-1: Key transformation**

We have noted initial 64-bit key is transformed into a 56-bit key by discarding every 8th bit of the initial key. Thus, for each a 56-bit key is available. From this 56-bit key, a different 48-bit Sub Key is generated during each round using a process called key transformation. For this, the 56-bit key is divided into two halves, each of 28 bits. These halves are circularly shifted left by one or two positions, depending on the round
For example, if the round numbers 1, 2, 9, or 16 the shift is done by only position for other rounds, the circular shift is done by two positions. The number of key bits shifted per round is shown in the figure.

| Round | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| #key bits shifted | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

**Figure -** number of key bits shifted per round

After an appropriate shift, 48 of the 56 bits are selected. for selecting 48 of the 56 bits the table is shown in the figure given below. For instance, after the shift, bit number 14 moves on the first position, bit number 17 moves on the second position, and so on. If we observe the table carefully, we will realize that it contains only 48-bit positions. Bit number 18 is discarded (we will not find it in the table), like 7 others, to reduce a 56-bit key to a 48-bit key. Since the key transformation process involves permutation as well as a selection of a 48-bit subset of the original 56-bit key it is called Compression Permutation.

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

**Figure -** compression permutation

Because of this compression permutation technique, a different subset of key bits is used in each round. That makes DES not easy to crack.

Step-2: Expansion Permutation
Recall that after initial permutation, we had two 32-bit plain text areas called Left Plain Text(LPT) and Right Plain Text(RPT). During the expansion permutation, the RPT is expanded from 32 bits to 48 bits. Bits are permuted as well hence called expansion permutation. This happens as the 32-bit RPT is divided into 8 blocks, with each block consisting of 4 bits. Then, each 4-bit block of the

previous step is then expanded to a corresponding 6-bit block, i.e., per 4-bit block, 2 more bits are added.

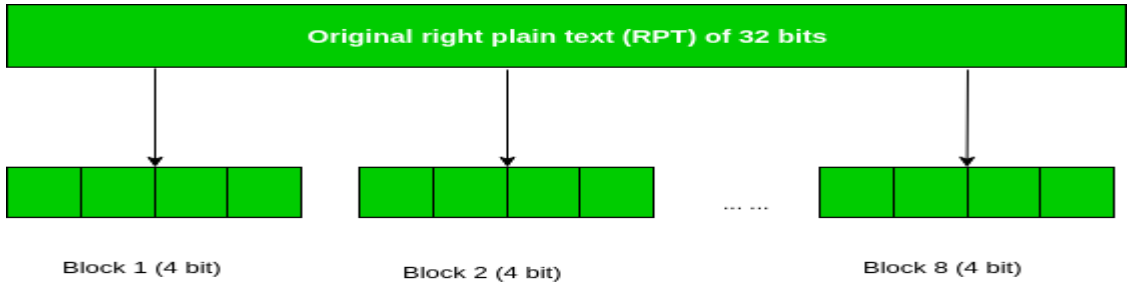



**Figure** - division of 32 bit RPT into 8 bit blocks

This process results in expansion as well as a permutation of the input bit while creating output. The key transformation process compresses the 56-bit key to 48 bits. Then the expansion permutation process expands the 32-bit RPT to 48-bits. Now the 48-bit key is XOR with 48-bit RPT and the resulting output is given to the next step, which is the S-Box substitution.

**Block Cipher and Public Key Cryptography**

**Block Cipher:**

A block cipher takes a block of plaintext bits and generates a block of ciphertext bits, generally of same size. The size of block is fixed in the given scheme. The choice of block size does not directly affect to the strength of encryption scheme. The strength of cipher depends up on the key length.

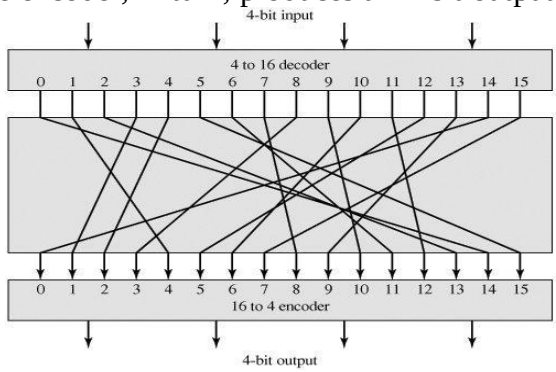
**Block Size:**

Block ciphers use a block of bits as the unit of encryption and decryption. To encrypt a 64-bit block, one has to take each of the $2^{64}$ input values and map it to one of the $2^{64}$ output values. The mapping should be one-to-one. Encryption and decryption operations of a block cipher are shown in Fig.

Some operations, such as permutation and substitution, are performed on the block of bits based on a key (a secret number) to produce another block of bits.

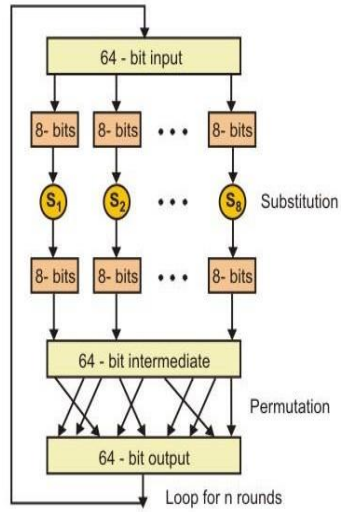
Permutation: The permutation is performed by a permutation box at the bit-level, which keeps the number of 0s and 1s same at the input and output. Although it can be implemented either by hardware or software, the hardware implementation is faster.

Substitution: Fig. shows the substitution is implemented with the help of three building blocks – a decoder, one p-box and an encoder. For an n-bit input, the decoder produces an 2n bit output having only one 1, which is applied to the P-box. The P-box permutes the output of the decoder and it is applied to the encoder. The encoder, in turn, produces an n-bit output. For example, if the input to the decoder is 011, the



output of the decoder is 00001000. Let the permuted output is 01000000, the output of the encoder is 011.

Most symmetric block ciphers are based on a Feistel Cipher Structure needed since must be able to decrypt ciphertext to recover messages efficiently. block ciphers look like an extremely It devid input




into 8-Bit pieces Substitute each 8-bit based on functions derived from the key. Permute the bitsbased on the key

Requires table of 264 entries for a 64-bit block

- Instead create from smaller building blocks
- 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
- S-P networks are based on the two primitive cryptographic operations we have seen before:
- substitution (S-box)
- permutation (P-box)
- provide confusion and diffusion of message
- diffusion – dissipates statistical structure of plaintext over bulk of ciphertext
- confusion – makes relationship between ciphertext and key as complex as possible

**Block Cipher Schemes:**

There is a vast number of block ciphers schemes that are in use. Many of them are publicly known. Most popular and prominent block ciphers are listed below.

Digital Encryption Standard (DES) − The popular block cipher of the 1990s. It is now considered as a 'broken' block cipher, due primarily to its small key size.

Triple DES − It is a variant scheme based on repeated DES applications. It is still a respected block ciphers but inefficient compared to the new faster block ciphers available.

Advanced Encryption Standard (AES) − It is a relatively new block cipher based on the encryption algorithm

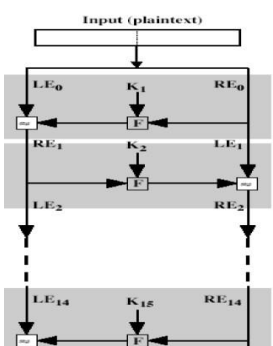
**Feistel cipher structure:**

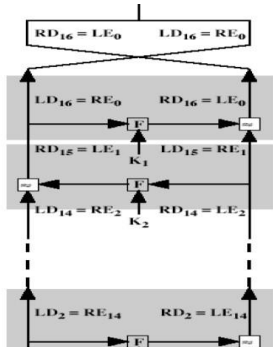
The plaintext block is divided into two halves L0 and R0. The two halves of the data pass through „n" rounds of processing and then combine to produce the ciphertext block.

Each round „i" has inputs Li-1 and Ri-1, derived from the previous round, as well as the subkey Ki, derived from the overall key K. in general, the subkeys Ki are different from K and from each other.

All rounds have the same structure. A substitution is performed on the left half of the data

This is done by applying a round function F to the right half of the data and then taking the XOR of the output of that function and the left half of the data.




The round function has the same general structure for each round but is parameterized by the round subkey ki. Following this substitution, a permutation is performed that consists of the interchange ofthe two halves of the data.

The process of decryption is essentially the same as the encryption process. The rule is as follows: use the cipher text as input to the algorithm, but use the subkey ki in reverse order. i.e., kn in the first round, kn-1 in second round and so on.

The exact realization of a Feistel network depends on the choice

of the following parameters and design features:

· Block size - Increasing size improves security, but slows cipher
· Key size - Increasing size improves security, makes exhaustive key searching harder, but may slow cipher
· Number of rounds - Increasing number improves security, but slows cipher
· Subkey generation - Greater complexity can make analysis harder, but slows cipher
· Round function - Greater complexity can make analysis harder, but slows cipher
· Fast software en/decryption & ease of analysis - are more recent concerns for practical use and testing.

The exact realization of a Feistel network depends on the choice of the following parameters and design features:

• Block size: Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion. Traditionally, a block size of 64 bits has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.

• Key size: Larger key size means greater security but may decrease encryption/decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.

• Number of rounds: The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.

• Subkey generation algorithm: Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.

• Round function F: Again, greater complexity generally means greater resistance to cryptanalysis. There are two other considerations in the design of a Feistel cipher:

• Fast software encryption/decryption: In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation.
Accordingly, the speed of execution of the algorithm becomes a concern.

• Ease of analysis: Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze. That is, if the algorithm can be concisely and clearly explained, it is easier to analyze that algorithm for cryptanalytic vulnerabilities and therefore develop a higher level of assurance as to its strength. DES, for example, does not have an easily analyzed functionality.

**Block Cipher Schemes**

There is a vast number of block ciphers schemes that are in use. Many of them are publicly known. Most popular and prominent block ciphers are listed below.

Digital Encryption Standard (DES) − The popular block cipher of the 1990s. It is now considered as a 'broken' block cipher, due primarily to its small key size.

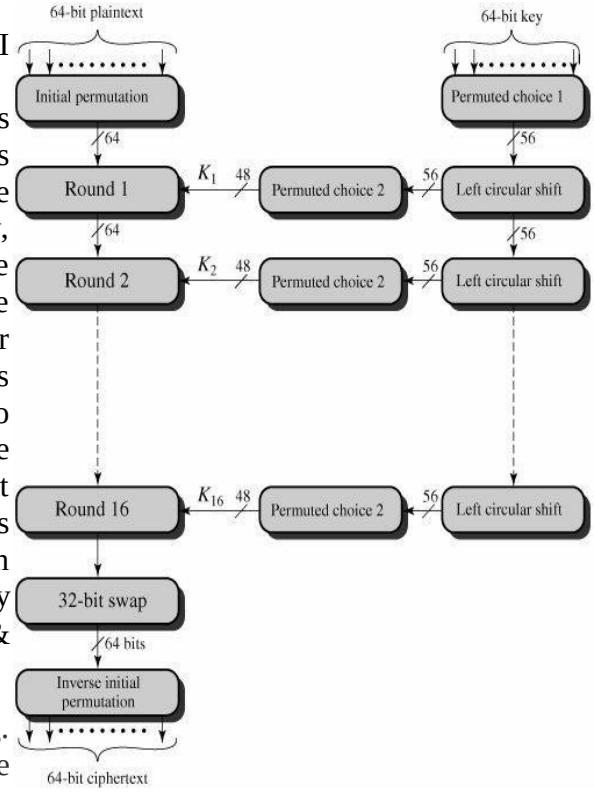
Triple DES − It is a variant scheme based on repeated DES applications. It is still a respected block ciphers but inefficient compared to the new faster block ciphers available.

Advanced Encryption Standard (AES) − It is a relatively new block cipher based on the encryption algorithm

**Data Encryption**

In May 1973, and again in Aug 1974 the NBS (now NIST) called for possible encryption algorithms for use in unclassified government applications response was mostly disappointing, however, IBM submitted their Lucifer design following a period of redesign and comment it became the Data Encryption Standard (DES) it was adopted as a (US) federal standard in Nov 76, published by NBS as a hardware only scheme in Jan 77 and by

ANSI for both hardware and software standards in ANSI X3.92-
1981 (also X3.106-1983 modes of use) subsequently it has been widely adopted and is now published in many standards around the world cf Australian Standard AS2805.5-1985 one of the largest users of the DES is the banking industry, particularly with EFT, and EFTPOS it is for this use that the DES has primarily been standardized, with ANSI having twice reconfirmed its recommended use for 5 year periods - a further extension is not expected however although the standard is public, the design criteria used are classified and have yet to be released there has been considerable controversy over the design, particularly in the choice of a 56-bit key, recent analysis has shown despite this that the choice was appropriate, and that DES is well designed, rapid advances in computing speed though have rendered the 56-bit key susceptible to exhaustive key search, as predicted by Diffie & Hellman.




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

Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases. First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the permuted input. This is followed by a phase consisting of 16 rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the preoutput.

Finally, the preoutput is passed through a permutation (IP-1) that is the inverse of the initial permutation function, to produce the 64-bit ciphertext. With the exception of the initial and final permutations, DES has the exact structure of a Feistel cipher.

## ➢Block Cipher Modes of Operation

Block cipher mode of operation is an algorithm that uses a block cipher to processes the data blocks of fixed size provide information security such as confidentiality or authenticity.the long message is divided into a series of sequential message blocks, and the cipher operates on these blocks one at a time.

A block cipher by itself is only suitable for the secure cryptographic transformation (encryption or decryption) of one fixed-length group of bits called a block.A mode of operation describes how to repeatedly apply a cipher's single-block operation to securely transform amounts of data larger than a block.

**Electronic Code Book (ECB) Mode**

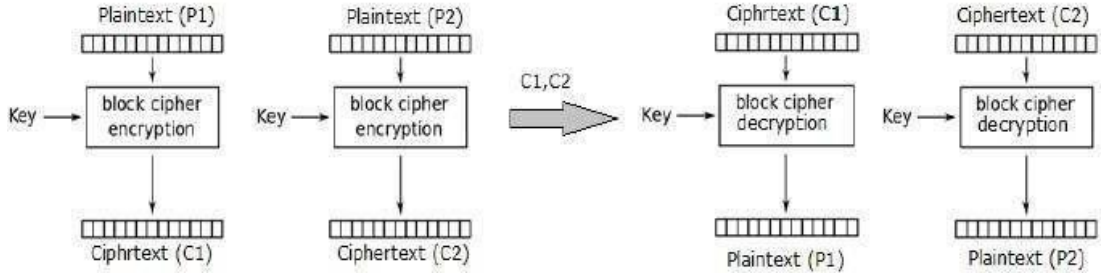
This mode is a most straightforward way of processing a series of sequentially listed message blocks.

Operation:

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

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

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




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

Analysis of ECB Mode:

In reality, any application data usually have partial information which can be guessed. For example, the range of salary can be guessed. A ciphertext from ECB can allow an attacker to guess the plaintext by trial-and-error if the plaintext message is within predictable.

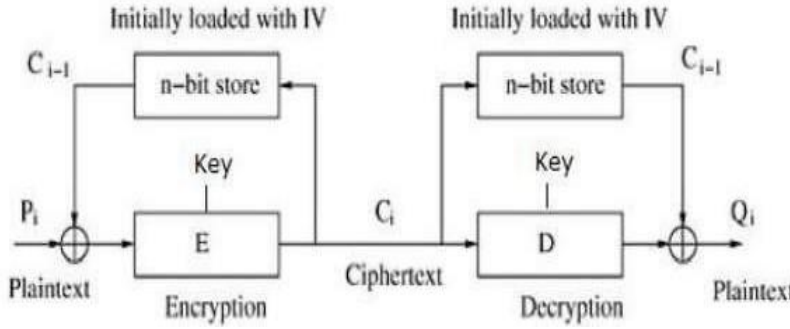
For example, if a ciphertext from the ECB mode is known to encrypt a salary figure, then a small number of trials will allow an attacker to recover the figure. In general, we do not wish to use a deterministic cipher, and hence the ECB mode should not be used in most applications.

**Cipher Block Chaining (CBC) Mode**

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

Operation:

The operation of CBC mode is depicted in the following illustration. The steps are as follows. Load the n-bit Initialization Vector (IV) in the top register. XOR the n-bit plaintext block with data value in top register. Encrypt the result of XOR operation with underlying block cipher with key K. Feed ciphertext block into top register and continue the operation till all plaintext blocks are processed.




For decryption, IV data is XORed with first ciphertext block decrypted. The first ciphertext block is also fed into to register replacing IV for decrypting next ciphertext block.

**Analysis of CBC Mode**

In CBC mode, the current plaintext block is added to the previous ciphertext block, and then the result is encrypted with the key. Decryption is thus the reverse process, which involves decrypting the current ciphertext and then adding the previous ciphertext block to the result.

Advantage of CBC over ECB is that changing IV results in different ciphertext for identical message. On the drawback side, the error in transmission gets propagated to few further blocks during decryption due to chaining effect.

It is worth mentioning that CBC mode forms the basis for a well-known data origin authentication mechanism. Thus, it has an advantage for those applications that require both symmetric encryption and data origin authentication.

**Cipher Feedback (CFB) Mode**

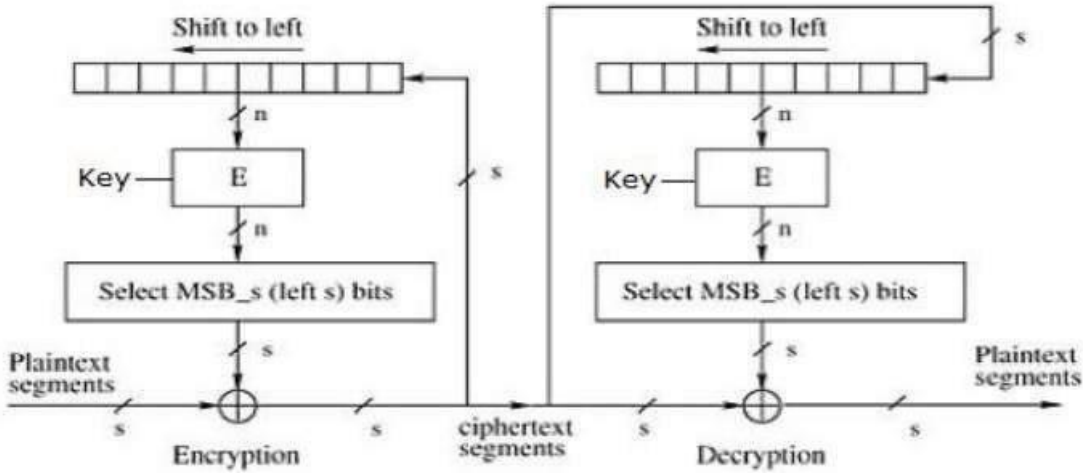
In this mode, each ciphertext block gets ‘fed back’ into the encryption process in order to encrypt the next plaintext block.

**Operation**

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

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

loaded at the start of decryption.




**Analysis of CFB Mode**

CFB mode differs significantly from ECB mode, the ciphertext corresponding to a given plaintext block depends not just on that plaintext block and the key, but also on the previous ciphertext block. In other words, the ciphertext block is dependent of message.

CFB has a very strange feature. In this mode, user decrypts the ciphertext using only the encryption process of the block cipher. The decryption algorithm of the underlying block cipher is never used.

Apparently, CFB mode is converting a block cipher into a type of stream cipher. The encryption algorithm is used as a key-stream generator to produce key-stream that is placed in the bottom register. This key stream is then XORed with the plaintext as in case of stream cipher.

By converting a block cipher into a stream cipher, CFB mode provides some of the advantageous properties of a stream cipher while retaining the advantageous properties of a block cipher. On the flip side, the error of transmission gets propagated due to changing of blocks.

**Output Feedback (OFB) Mode**

It involves feeding the successive output blocks from the underlying block cipher back to it. These feedback blocks provide string of bits to feed the encryption algorithm which act as the key- stream generator as in case of CFB mode. The key stream generated is XOR-ed with the plaintext sblocks. The OFB mode requires an IV as the initial random n-bit input block. The IV need not be secret. The operation is depicted in the following illustration

**Counter (CTR) Mode:**

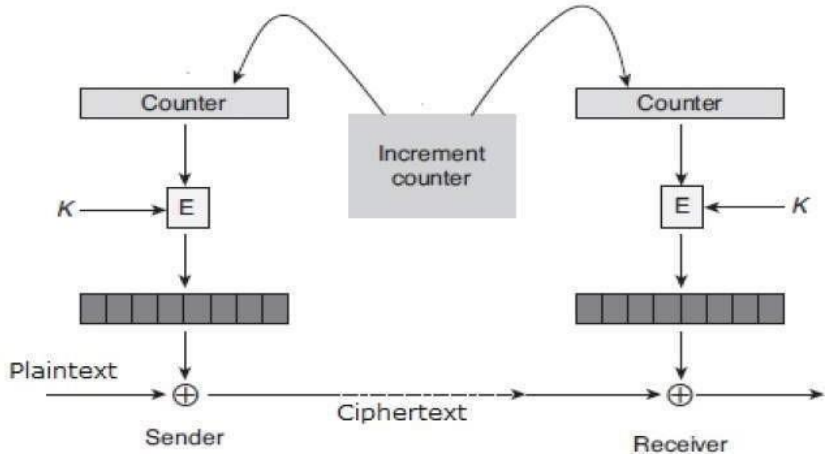
It can be considered as a counter-based version of CFB mode without the feedback. In this mode, both the sender and receiver need to access to a reliable counter, which computes a new shared value each time a ciphertext block is exchanged. This shared counter is not necessarily a secret value, but challenge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

Operation

Both encryption and decryption in CTR mode are depicted in the following illustration. Steps in operation are, Load the initial counter value in the top register is the same for both the sender and the receiver. It plays the same role as the IV in CFB (and CBC) mode.

Encrypt the contents of the counter with the key and place the result in the bottom register. Take the first plaintext block P1 and XOR this to the contents of the bottom register. The result of this is C1. Send C1 to

the receiver and update the counter. The counter update replaces the ciphertext feedback in CFB mode. Continue in this manner until the last plaintext block has been encrypted. The decryption is the reverse process. The ciphertext block is XORed with the output of encrypted contents of counter value. After decryption of each ciphertext block counter is updated as in case of encryption.




Analysis of Counter Mode

It does not have message dependency and hence a ciphertext block does not depend on the previous plaintext blocks. Like CFB mode, CTR mode does not involve the decryption process of the block cipher. This is because the CTR mode is really using the block cipher to generate a key-stream, which is encrypted using the XOR function. In other words, CTR mode also converts a block cipher to a streamcipher.

The serious disadvantage of CTR mode is that it requires a synchronous counter at sender and receiver. Loss of synchronization leads to incorrect recovery of plaintext. However, CTR mode has almost all advantages of CFB mode. In addition, it does not propagate error of transmission at all.

## ➢ Conventional Encryption Algorithms

**Conventional Encryption** involves transforming plaintext messages into ciphertext messages that are decrypted only by the intended receiver. Both sender and receiver agree upon a secrete key to be us encrypting and decrypting. Usually the secrete key is transmitted via public key encryption methods.

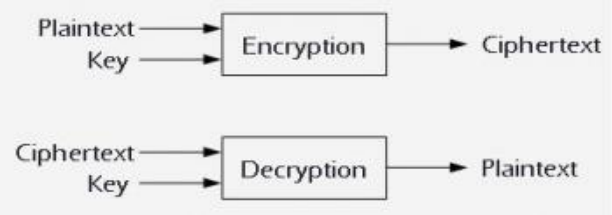



Figure 1: Flow Diagram

In conventional encryption, it is assumed that mathematically impossible to derive the plaintext fro ciphertext without the key.Therefore, it is essential th key remains secret.

These encryption algorithms are used in practice due to their efficiency in encrypting/decryptin these algorithms have vulnerabilities. One aspect of these vulnerabilities is the total number of keys availa choose from. Larger key domains reduce possibility of brute force attacks. The key length is another asp these vulnerabilities since they will produce periodic patterns in the ciphertext. Longer keys often r periodicity. The goal of conventional encryption algorithms is to produce truly randomized ciphertexts, suc the use of frequency analysis on individual ciphertext symbols or ciphertext blocks is useless.

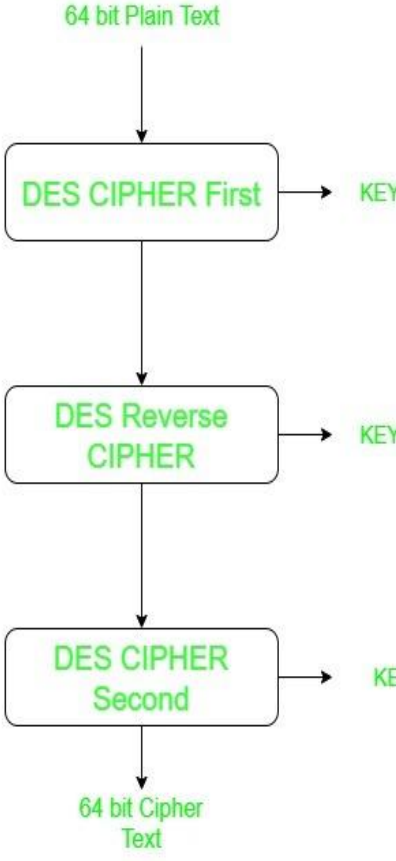
## ➢ Triples DES

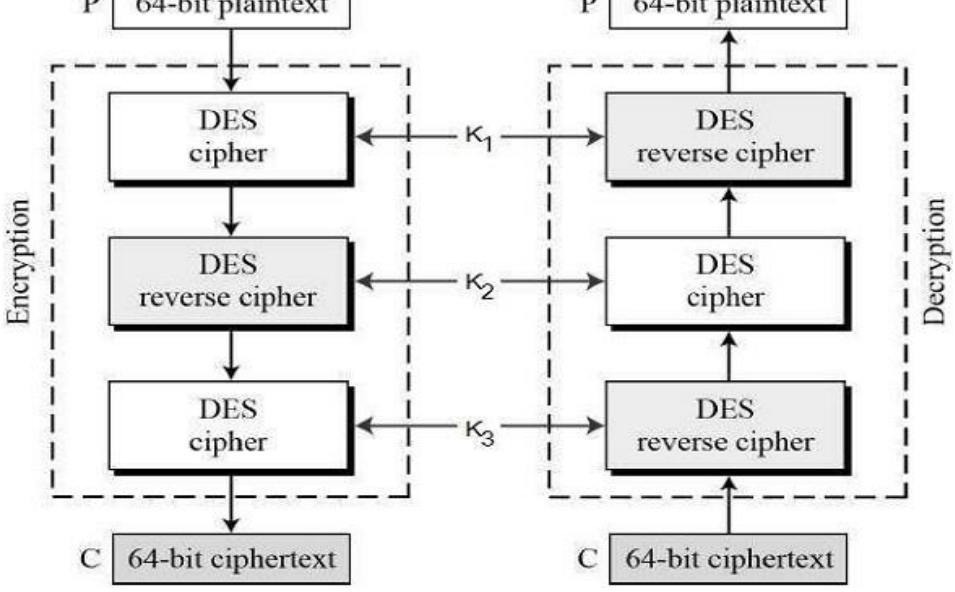
Triple DES − It is a variant scheme based on repeated DES applications. It is still a respected block ciphe inefficient compared to the new faster block ciphers available.

Triple Data Encryption Standard (DES) is a type of computerized cryptography where block cipher algorithms are applied three times to each data block. The key size is increased in Triple DES to ensure additional security through encryption capabilities. Each block contains 64 bits of data. Three keys are referred to as bundle keys with 56 bits per key. There are three keying options in data encryption standards:

1. All keys being independent
2. Key 1 and key 2 being independent keys
3. All three keys being identical

Key option #3 is known as triple DES. The triple DES key length contains 168 bits but the key security falls to 112 bits.






The encryption scheme illustrated as follows −

The encryption-decryption pr is as follows −

- Encrypt the pla blocks using single DES wit $K_1$.
- Now decrypt the outp step 1 using single DES wit $K_2$.
- Finally, encrypt the o of step 2 using single DES key $K_3$.
- The output of step 3 ciphertext.

- Decryption of a ciphertext is a reverse process. User first decrypt using $K_3$, then encrypt with K finally decrypt with $K_1$.

Due to this design of Triple DES as an encrypt–decrypt–encrypt process, it is possible to use a 3T (hardware) implementation for single DES by setting $K_1$, $K_2$, and $K_3$ to be the same value. This pro backwards compatibility with DES.

Second variant of Triple DES (2TDES) is identical to 3TDES except that $K_3$is replaced by $K_1$. In other w user encrypt plaintext blocks with key $K_1$, then decrypt with key $K_2$, and finally encrypt with $K_1$ again. Ther 2TDES has a key length of 112 bits.

Triple DES systems are significantly more secure than single DES, but these are clearly a much slower pr than encryption using single DES.

## ➢ International Data Encryption Algorithm

In cryptography, the **International Data Encryption Algorithm** (**IDEA**), originally called **Improved Prop Encryption Standard** (**IPES**), is a symmetric-key block cipher In cryptography, block ciphers are important in the designing of many cryptographic algorithms and are widely used to encrypt the bulk of d chunks. By chunks, it means that the cipher takes a fixed size of the plaintext in the encryption proces generates a fixed size ciphertext using a fixed-length key. An algorithm's strength is determined by it length. The Simplified **International Data Encryption Algorithm (IDEA)** is a **symmetric key block** c that

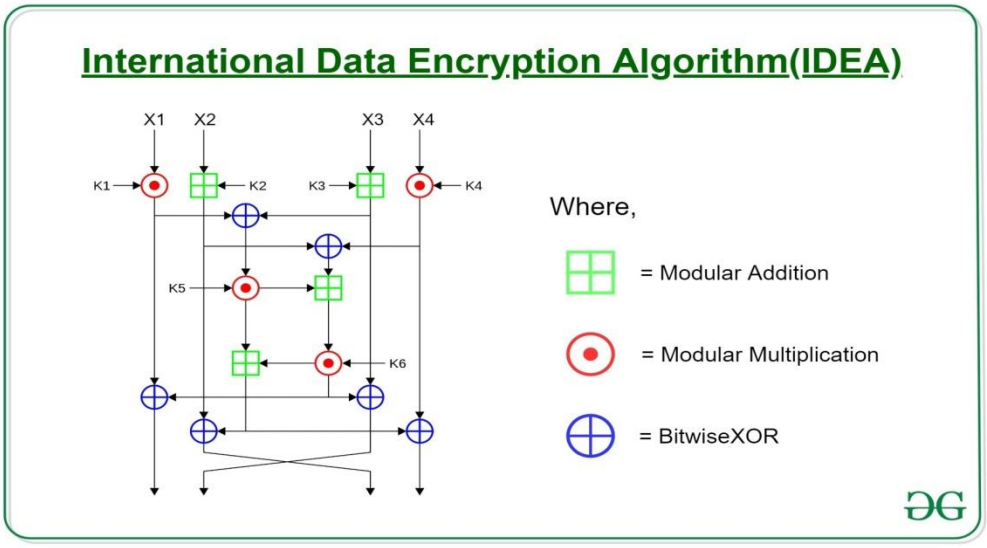
- uses a fixed-length plaintext of **16 bits** and
- encrypts them in **4 chunks of 4 bits** each
- to produce **16 bits ciphertext**.
- The length of the key used is **32 bits**.
- The key is also divided into 8 blocks of 4 bits each.

This algorithm involves a series of 4 identical complete rounds and 1 half-round. Each complete rou involves a series of 14 steps that includes operations like:

- Bitwise XOR
- Addition modulo
- Multiplication modulo +1

After 4 complete rounds, the final "half-round" consists of only the first 4 out of the 14 steps previ used in the full rounds. To perform these rounds, each binary notation must be converted to its equiv decimal notation, perform the operation and the result obtained should be converted back to the b representation for the final result of that particular step.

**Key Schedule:** 6 subkeys of 4 bits out of the 8 subkeys are used in each complete round, while 4 are used half-round. So, 4.5 rounds require 28 subkeys. The given key, 'K', directly gives the first 8 subkeys. By ro the main key left by 6 bits between each group of 8, further groups of 8 subkeys are created, implying less one rotation per round for the key (3 rotations).




| | K1 | K2 | K3 | K4 | K5 | K6 |
|---|---|---|---|---|---|---|
| Round 1 | 1101 | 1100 | 0110 | 1111 | 0011 | 1111 |
| Round 2 | 0101 | 1001* | 0001 | 1011 | 1100 | 1111 |
| Round 3 | 1101 | 0110 | 0111 | 0111* | 1111 | 0011 |
| Round 4 | 1111 | 0101 | 1001 | 1101 | 1100 | 0110* |
| Round 4.5 | 1111 | 1101 | 0110 | 0111 | | |

* denotes a shift of bits

Notations used in the 14 steps:

| Symbol | Operation |
|---|---|
| * | Multiplication modulo +1 |
| + | Addition modulo |
| ^ | Bitwise XOR |

The 16-bit plaintext can be represented as **X1 || X2 || X3 || X4**, each of size 4 bits. The 32-bit k broken into 8 subkeys denoted as K1 || K2 || K3 || K4 || K5 || K6 || K7 || K8, again of size 4 bits Each round of 14 steps uses the three algebraic operation-Addition modulo (2^4), Multiplic modulo (2^4)+1 and Bitwise XOR. The steps involved are as foll

1. X1 * K1
2. X2 + K2
3. X3 + K3

4. X4 * K4
5. Step 1 ^ Step 3
6. Step 2 ^ Step 4
7. Step 5 * K5
8. Step 6 + Step 7
9. Step 8 * K6
10. Step 7 + Step 9
11. Step 1 ^ Step 9
12. Step 3 ^ Step 9
13. Step 2 ^ Step 10
14. Step 4 ^ Step 10

The input to the next round is Step 11 || Step 13 || Step 12 || Step 14, which becomes X1 || X2 || X3 |
This swap between 12 and 13 takes place after each complete round, except the last complete round
round), where the input to the final half round is Step 11 || Step 12 || Step 13 || Step 14.
After last complete round, the half-round is as foll

1. X1 * K1
2. X2 + K2
3. X3 + K3
4. X4 * K4

The final output is obtained by concatenating the blocks.

**Example:**

**Key:** 1101 1100 0110 1111 0011 1111 0101 1001
**Plaintext:** 1001 1100 1010 1100
**Ciphertext:** 1011 1011 0100 1011

**Explanation:**

The explanation is only for 1st complete round (the remaining can be implemented similarly) and th
last half-round.

**Round 1:**

- From the plaintext: **X1 – 1001, X2 – 1100, X3 – 1010, X4 – 1100**

From the table above: **K1 – 1101, K2 – 1100, K3 – 0110, K4 – 1111, K5 – 0011, K6 – 1111**

(1001(9) * 1101(13))(mod 17) = 1111(15)
(1100(12) + 1100(12))(mod 16) = 1000(8)
(1010(10) + 0110(6))(mod 16) = 0000(0)

(1100(12) * 1111(15))(mod 17) = 1010(10)

(1111(15) ^ 0000(0)) = 1111(15)

(1000(8) ^ 1010(10)) = 0010(2)

(1111(15) * 0011(3))(mod 17) = 1011(11)

(0010(2) + 1011(11))(mod 16) = 1101(13)

(1101(13) * 1111(15))(mod 17) = 1000(8)

(1011(11) + 1000(8))(mod 16) = 0011(3)

(1000(8) ^ 1111(15)) = 0111(7)

(1000(8) ^ 0000(0)) = 1000(8)

(0011(3) ^ 1000(8)) = 1011(11)

(0011(3) ^ 1010(10)) = 1001(9)

- **Round 1 Output**: 0111 1011 1000 1001 (Step 12 and Step 13 results are interchanged*)*
- **Round 2:**
- From Round 1 output: **X1 – 0111, X2 – 1011, X3 – 1000, X4 – 1001**

- From the table above: **K1 – 0101, K2 – 1001, K3 – 0001, K4 – 1011, K5 – 1100, K6 1111**

- **Round 2 Output**: 0110 0110 1110 1100 *(Step 12 and Step 13 results are interchange*
- **Round 3:**
- From Round 2 Output: **X1 – 0110, X2 – 0110, X3 – 1110, X4 – 1100**

- From the table above: **K1 – 1101, K2 – 0110, K3 – 0111, K4 – 0111, K5 – 1111, K6 0011**

- **Round 3 Output**: 0100 1110 1011 0010 *(Step 12 and Step 13 results are interchange*
- **Round 4:**
- From Round 3 Output: **X1 – 0100, X2 – 1110, X3 – 1011, X4 – 0010**

- From the table above: **K1 – 1111, K2 – 0101, K3 – 1001, K4 – 1101, K5 – 1100, K6 0110**

- **Round 4 Output**: 0011 1110 1110 0100 *(Step 12 and Step 13 results are interchange*
- **Round 4.5:**
- From Round 4 Output: **X1 – 0011, X2 – 1110, X3 – 1110, X4 – 0100**

- From the table above: **K1 – 1111, K2 – 1101, K3 – 0110, K4 – 0111**

- **Round 4.5 Output**: 1011 1011 0100 1011 (Step 2 and Step 3 results are **not** interchanged)

(0011(3) * 1111(15))(mod 17) = 1011(11)

(1110(14) + 1101(13))(mod 16) = 1011(11)

(1110(14) + 0110(6))(mod 16) = 0100(4)

(0100(4) * 0111(7))(mod 17) = 1011(11)

- **Final Ciphertext** is **1011 1011 0100 1011**

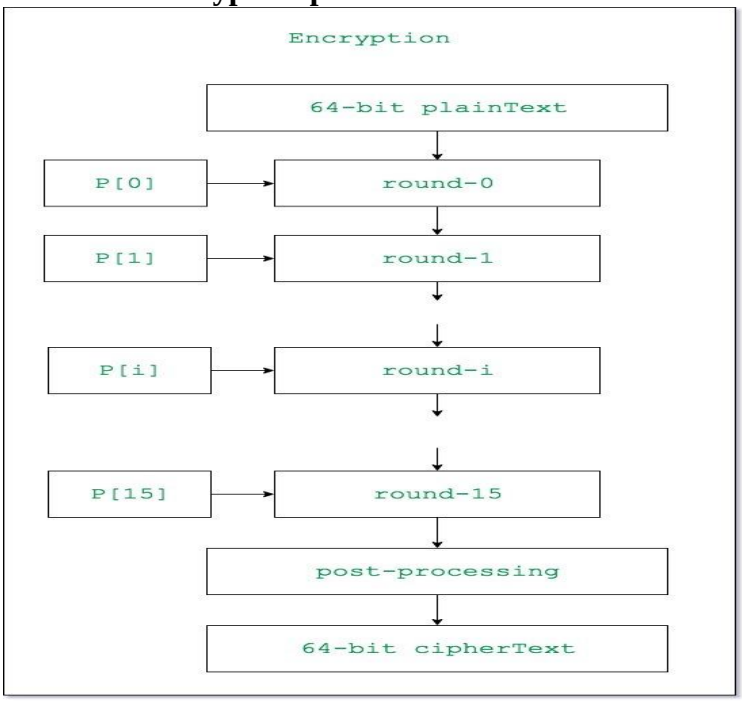
## ➢ Blowfish

**Blowfish** is an encryption technique designed by **Bruce Schneier** in 1993 as an alternative to DES Encry Technique. It is significantly faster than DES and provides a good encryption rate with effective cryptanalysis technique found to date. It is one of the first, secure block cyphers not subject t patents and hence freely available for anyone to use.

1. block Size: 64-bits
2. key Size: 32-bits to 448-bits variable size
3. number of subkeys: 18 [P-array]
4. number of rounds: 16
5. number of substitution boxes: 4 [each having 512 entries of 32-bits each]

Blowfish Encryption Algorithm

**The entire encryption process can be elaborated as:**




Let's see each step one by one:

**Step1: Generation of subkeys:**

- 18 sub keys{P[0]…P[17]} are needed in both encryption as well as decryption process and the subkeys are used for both the processes.
- These 18 subkeys are stored in a P-array with each array element being a 32-bit entry.
- It is initialized with the digits of pi(?).
- The hexadecimal representation of each of the subkeys is given by:

P[0] = "243f6a88"
P[1] = "85a308d3"
.
.
.
P[17] = "8979fb1b"

**32-bit hexadecimal representation of initial values of sub-keys**

| | | | |
|---|---|---|---|
| P[0] | : 243f6a88 | P[9] | : 38d01377 |
| P[1] | : 85a308d3 | P[10] | : be5466cf |
| P[2] | : 13198a2e | P[11] | : 34e90c6c |
| P[3] | : 03707344 | P[12] | : c0ac29b7 |
| P[4] | : a4093822 | P[13] | : c97c50dd |
| P[5] | : 299f31d0 | P[14] | : 3f84d5b5 |
| P[6] | : 082efa98 | P[15] | : b5470917 |
| P[7] | : ec4e6c89 | P[16] | : 9216d5d9 |
| P[8] | : 452821e6 | P[17] | : 8979fb1b |

- Now each of the subkey is changed with respect to the input key as:

P[0] = P[0] xor 1st 32-bits of input key

P[1] = P[1] xor 2nd 32-bits of input key

.
.
.
P[i] = P[i] xor (i+1)th 32-bits of input key

(roll over to 1st 32-bits depending on the key length)

.
.
.

P[17] = P[17] xor 18th 32-bits of input key
(roll over to 1st 32-bits depending on key length)
The resultant P-array holds 18 subkeys that is used during the entire encryption process

**Step2: initialise Substitution Boxes:**

- 4 Substitution boxes(S-boxes) are needed{S[0]…S[4]} in both encryption as well as decryption pr

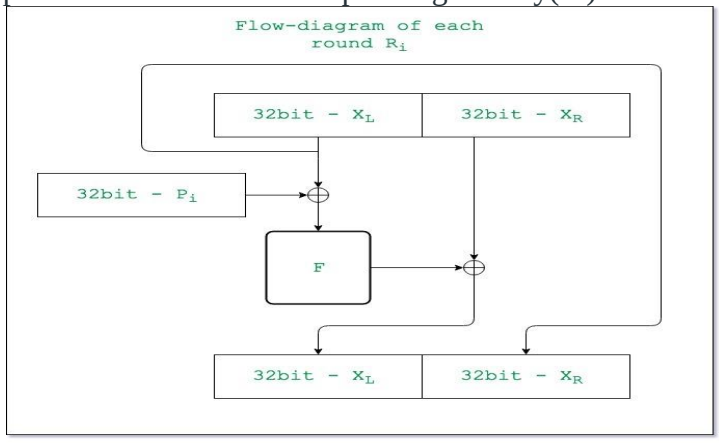
with each S-box having 256 entries{S[i][0]…S[i][255], 0&lei&le4} where each entry is 32-bit.

- It is initialized with the digits of pi(?) after initializing the P-array.

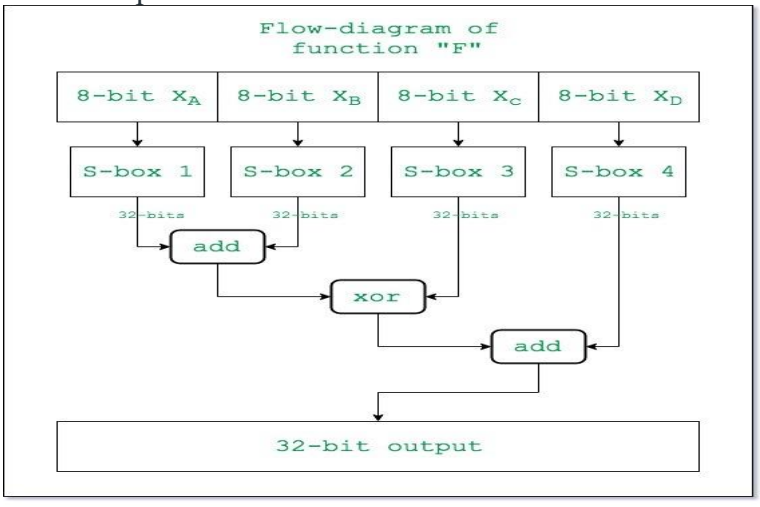
**Step3: Encryption:**

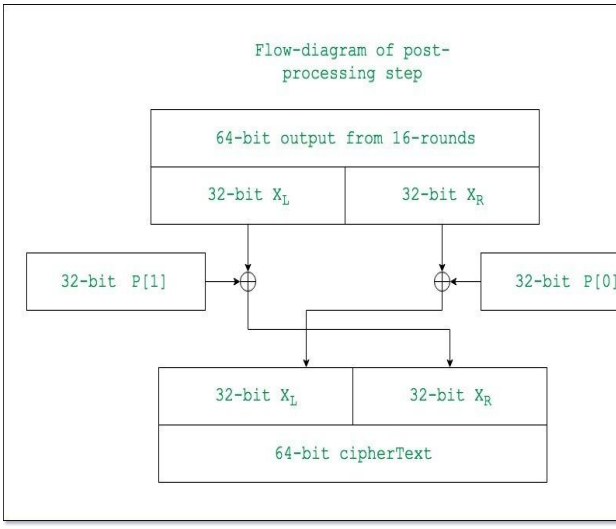
- The encryption function consists of two parts:

**a. Rounds:** The encryption consists of 16 rounds with each round (Ri) taking inputs the plainText(P.T.) fr
previous round and corresponding subkey(Pi). The description of each round is as follows:




The description of the function ” F ” is as follows:



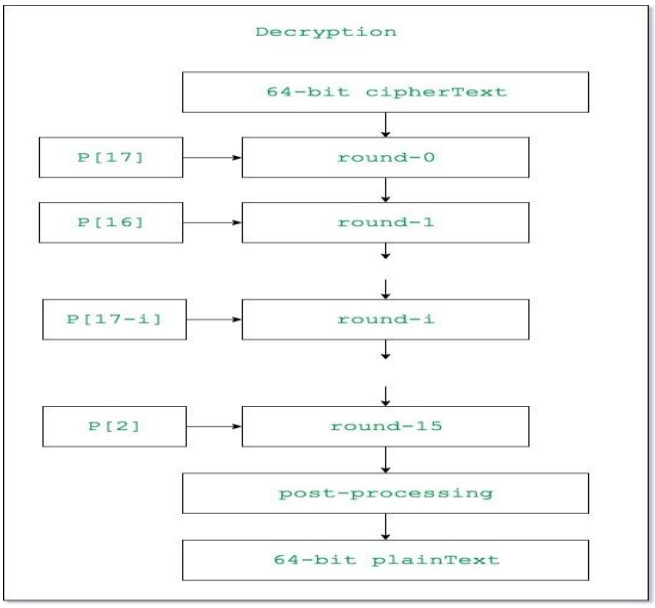




Here the function “add” is addition modulo 2^32.

**b. Post-processing:** The output after the 16 rounds is processed as follows:

**Decryption**

The decryption process is similar to that of encryption and the subkeys are used in reverse {P[17] – P[0]}
entire decryption process can be elaborated as:




Let's see each step one by one:

**Step1: Generation of subkeys:**

- 18 subkeys {P[0]…P[17]} are needed in decryption process.
- These 18 subkeys are stored in a P-array with each array element being a 32-bit entry.
- It is initialized with the digits of pi(?).
- The hexadecimal representation of each of the subkeys is given by:

P[0] = "243f6a88"
P[1] = "85a308d3"
.
.
.
P[17] = "8979fb1b"
P[0] = P[0] xor 1st 32-bits of input key

P[1] = P[1] xor 2nd 32-bits of input key
.
.
.
P[i] = P[i] xor (i+1) th 32-bits of input key
(roll over to 1st 32-bits depending on the key length)
.
.
.
P [17] = P[17] xor 18th 32-bits of input key
(roll over to 1st 32-bits depending on key length)

**The resultant P-array holds 18 subkeys that is used during the entire encryption process**

**Step2: initialise Substitution Boxes:**

- 4 Substitution boxes(S-boxes) are needed{S[0]…S[4]} in both encryption as well as decryption pr

with each S-box having 256 entries{S[i][0]…S[i][255], 0&lei&le4} where each entry is 32-bit.

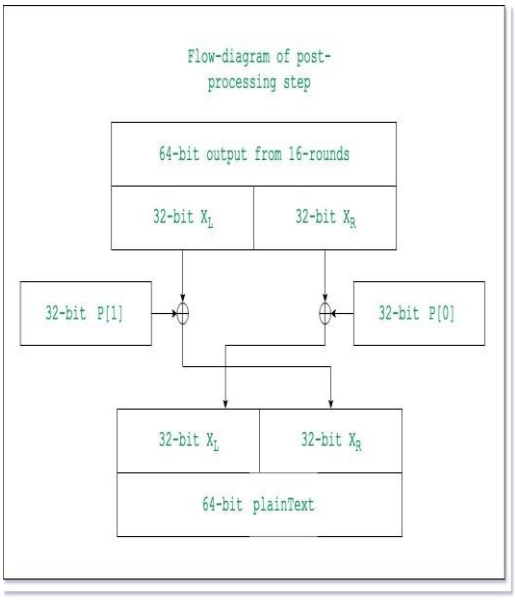
- It is initialised with the digits of pi(?) after initializing the P-array.

**Step3: Decryption:**

- The Decryption function also consists of two parts:

1. **Rounds:** The decryption also consists of 16 rounds with each round(Ri)(as explained abov

taking inputs the cipherText(C.T.) from previous round and corresponding subkey(P[17-i])(i.e for decrypt the subkeys are used in reverse).




2. **Post-processing:** The output after the 16 rounds i processed as follows:

**Note:** See encryption for the initial values of P-array.

- Now each of the subkeys is changed with respect to the i key as:

**Advantages and Disadvantages of Blowfish Algorithm:**

- Blowfish is a fast block cipher except when changing key Each new key requires a pre-processing equivalent to 4KB of te
- It is faster and much better than DES Encryption.
- Blowfish uses a 64-bit block size which makes it vulnera to birthday attacks.
- A reduced round variant of blowfish is known to be susceptible to known plain text attacks (2nd order differential attacks – 4 rounds).

**Applications of Blowfish Algorithm:**

- Bulk Encryption.
- Packet Encryption(ATM Packets)
- Password Hashing

## ➢ RC5

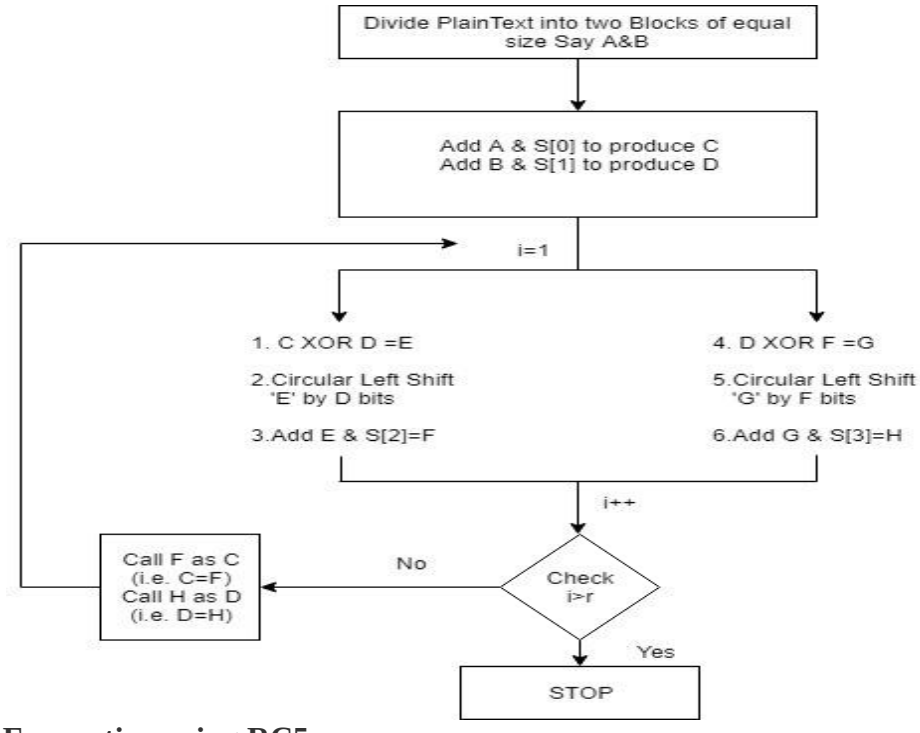
In cryptography, **RC5** is a symmetric-key block cipher notable for its simplicity. Designed by R Rivest in 1994, *RC* stands for "Rivest Cipher", or alternatively, "Ron's Code" (compare RC2 and RC4).

Unlike many schemes, RC5 has a variable block size (32, 64 or 128 bits), key size (0 to 2040 bits number of rounds (0 to 255). The original suggested choice of parameters was a block size of 64 bits, a 12 key and 12 rounds.

A key feature of RC5 is the use of data-dependent rotations; one of the goals of RC5 was to prom study and evaluation of such operations as a cryptographic primitive. RC5 also consists of a nu of modular additions and eXclusive OR (XOR)s. The general structure of the algorithm is a Feistel-like net The encryption and decryption routines can be specified in a few lines of code. The key schedule, howev more complex, expanding the key using an essentially one-way function with the binary expansion both e and the golden ratio as sources of "nothing up my sleeve numbers". The tantalising simplicity o algorithm together with the novelty of the data-dependent rotations has made RC5 an attractive object of for cryptanalysts. The RC5 is basically denoted as RC5-w/r/b where w=word size in bits, r=number of ro b=number of 8-bit bytes in the key.

## RC-5 Algorithm

- In RC-5, the word size (i.e. input plaintext block size), number of rounds and number of keys are not fixed i.e. all can be of variable length.
- Once w, r, k (word size, number of rounds, number of keys) are finalized then they remain same for the rounds.
- Plain text can be 32 bits, 64 bits or 128 bits
- Number of rounds can be between 0-255
- Key size can be between 0 to 255 bytes




**Encryption using RC5**

Encryption involved several rounds of a simple function. 12 or 20 rounds seem to be recommended, depe on security needs and time considerations.

We initialize the counter to 1 and perform some permutation and combination using addition and XOR

The algorithm works into two phases:

a. First it starts with phase one

b. Output of phase one become input of phase two

We divide the plaintext block into two equal parts A and B

Then they are XOR with two subkeys S{0} and S{1}

C=A+S[0]C=A+S[0]

D=B+S[1]D=B+S[1]

for i = 1 to r do:

1. C ⊕ D = E
2. perform circular left shift on E by D bits
3. add E and S[2 * i] and store the result in F which is input for step 4
4. D ⊕ F = G
5. perform circular left shift on G by F bits
6. add G and S[2 * i + 1] and store the result in H
7. If i< r

Call F as C and H as D and repeat the steps from 1 to 7

else stop

- Once both the phases are completed the counter is incriminated and we check if it is greater than the number of rounds, if yes then the algorithm terminals and if no then the algorithm iterates.

**Decryption:**

Decryption is a straightforward reversal of the encryption process

**Example:**

```
Key : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Plain Text : 00000000 00000000
Cipher Text : EEDBA521 6D8F4B15
```

RC5 is a block cipher and addresses two word blocks at a
Depending on input plain text block size, number of rounds and key size, various instances of RC5 c
defined and each instance is denoted as RC5-w/r/b where w=word size in bits, r=number of rounds and b
size in bytes.
Allowed values are:

| Parameter | Possible Value |
|---|---|
| block/word size (bits) | 16, 32, 64 |
| Number of Rounds | 0 – 255 |
| Key Size (bytes) | 0 – 255 |

**Note –** Since at a time, RC5 uses 2 word blocks, the plain text block size can be 32, 64 or 128 bits.
Notation used in the algorithm:

| Symbol | Operation |
|---|---|
| x <<< y | Cyclic left shift of x by y bits |
| + | Two's complement addition of words where addition is modulo $2^w$ |
| ^ | Bit wise Exclusive-OR |

**Step-1:** Initialization of constants P and Q
RC5 makes use of 2 magic constants P and Q whose value is defined by the word size w.

| Word Size (bits) | P (Hexadecimal) | Q (Hexadecimal) |
|---|---|---|
| 16 | b7e1 | 9e37 |
| 32 | b7e15163 | 9e3779b9 |
| 64 | b7e151628aed2a6b | 9e3779b97f4a7c15 |

For any other word size, P and Q can be determined as:

```
P = Odd((e-2)2^w)
Q = Odd((ϕ-2)2^w)
```

Here, Odd(x) is the odd integer nearest to x, e is the base of natural logarithms and $\phi$ is the golden ratio.
**Step-2:** Converting secret key K from bytes to words.
Secret key K of size b bytes is used to initialize array L consisting of c words where c = b/u, u = w/8 and w =
word size used for that particular instance of RC5. For example, if we choose w=32 bits and Key k is of size
bytes then, u=32/8=4, c=b/u=96/4=24.
L is pre initialized to 0 value before adding secret key K to it.

```
for i=b-1 to 0
    L[i/u] = (L[u/i] <<< 8) + K[i]
```

**Step-3:** Initializing sub-key S
Sub-key S of size t=2(r+1) is initialized using magic constants P and Q.

```
S[0] = P
for i = 1 to 2(r+1)-1
```

```
    S[i] = S[i-1] + Q)
```

**Step-4:** Sub-key mixing

The RC5 encryption algorithm uses Sub key S. L is merely, a temporary array formed on the basis of user entered secret key

Mix in user's secret key with S and L.

```
i = j = 0
A = B = 0
do 3 * max(t, c) times:
    A = S[i] = (S[i] + A + B) <<< 3
    B = L[j] = (L[j] + A + B) <<< (A + B)
    i = (i + 1) % t
    j = (j + 1) % c
```

**Step-5:** Encryption.

We divide the input plain text block into two registers A and B each of size w bits. After undergoin encryption process the result of A and B together forms the cipher text block.

RC5 Encryption Algorithm:

1. One time initialization of plain text blocks A and B by adding S[0] and S[1] to A and B respect These operations are mod $2^w$.
2. XOR A and B. A=A^B
3. Cyclic left shift new value of A by B bits.
4. Add S[2*i] to the output of previous step. This is the new value of A.
5. XOR B with new value of A and store in B.
6. Cyclic left shift new value of B by A bits.
7. Add S[2*i+1] to the output of previous step. This is the new value of B.
8. Repeat entire procedure (except one time initialization) r times.

```
A = A + S[0]
B = B + S[1]
for i = 1 to r do:
   A = ((A ^ B) <<< B) + S[2 * i]
   B = ((B ^ A) <<< A) + S[2 * i + 1]
return A, B
```

Alternatively, RC5 Decryption can be defined as:

```
for i = r down to 1 do:
   B = ((B - S[2 * i + 1]) >>> A) ^ A
   A = ((A - S[2 * i]) >>> B) ^ B
B = B - S[1]
A = A - S[0]
return A, B .
```

## ➢ Characteristics of advanced symmetric Block Ciphers:

Symmetric Block Cipher FactsA b
cipher takes a fixed-length number of bits, referred to as a block, and encrypts them all at once. Be aware of th
following facts regarding symmetric key block ciphers:

- Block ciphers are fast.
- Block ciphers can process large amounts of data. They do not process small amounts of data well.
- Block ciphers use a substitution and transposition function.
- A *round* refers to data going through one substitution and transposition process.
- Large amounts of data processed through a block cipher may begin to show patterns in the cipher text.

Advanced Encryption Standard (AES) is an iterative symmetric key block cipher that was developed as a replacement to DES in 2001. AES:

- Uses the Rijndael Block Cipher which is resistant to all known attacks.
- Uses a variable-length block and key length (128-, 192-, or 256-bit keys).

Block cipher is an algorithm that operates on bits called block.
The characteristics of Advanced symmetric block ciphers are -
1) **Key dependent S-boxes** - S-box performs substitution and depicts the relationship between key and cipher
2) **Data dependent rotation** - Data dependent rotation results in differences in the amount of rotation.
3) **Variable plain text** or cipher text block length
4) **Operations on both plain and ciphered data**

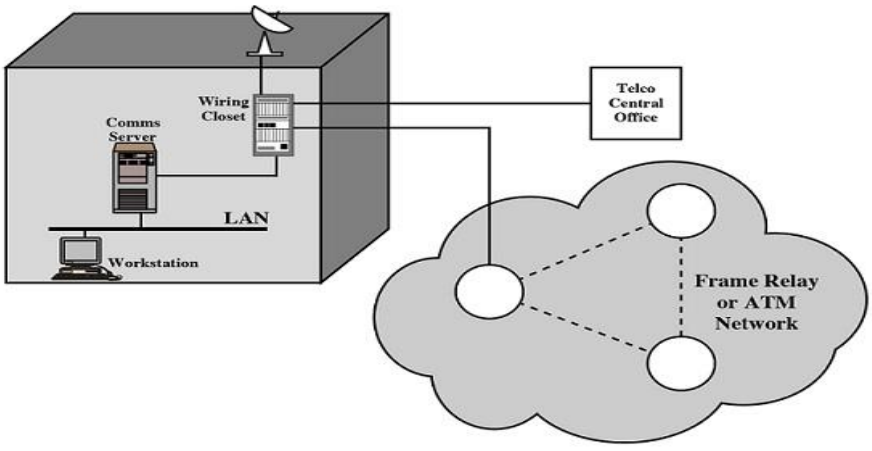
## ➢ Confidentiality Using Conventional Encryption

### ➢ Confidentiality

- Providing confidentiality through the use of secret-key encryption has historically been the focus of cryptology.
- This topic remains important in itself, though other considerations have emerged in the last few decad
- An understanding of the issues involved here clarifies those in other applications of encryption andh motivate the development of public-key encryption.

**Placement of Encryption Function**
Issues involved:
What should be encrypted?
Where should encryption be done?
Two approaches :
Encryption
End-to-end encryption
To make the decisions, one should first examine thepotential locations of security attacks.

## Points of Vulnerability

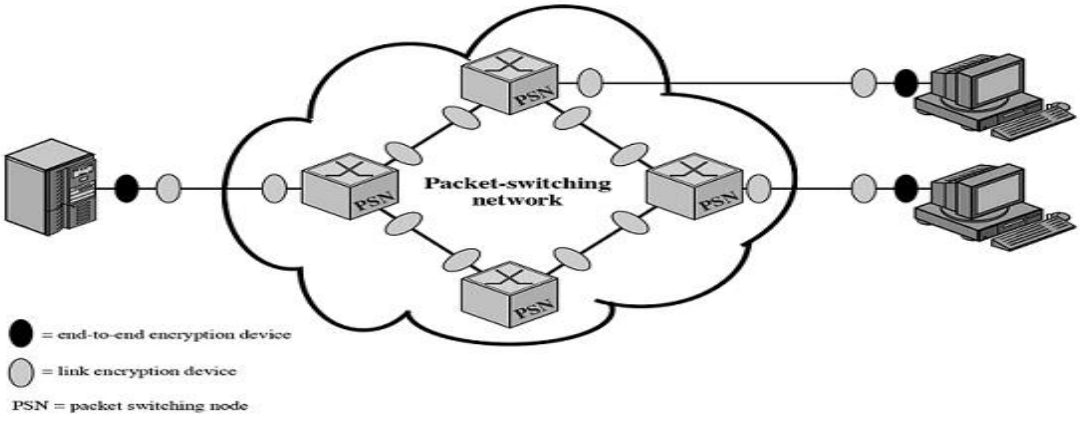



## Locations for Confidentiality Attacks

Consider a user workstation in a typical business organization. The points of vulnerability include:

- The LAN that the workstation is attached to: *eavesdropping* on the LAN, which is typically a *broadcast* network.
- The Wiring closet: tapping the wires.
- Communications links out of the Wiring closet: invasive or inductive tapping.
- Processors along the path to the outside: modifying the hardware or software, etc.

## Encryption in Packet-Switching Networks




**Link Encryption**

- Each vulnerable communications link is equipped onboth ends with an encryption device. Thus, al over all communications links is secured.
- The message must be decrypted each time it enters a packet switch. Thus, the message is vulnerable switch.

- Many keys must be provided. However, each key needsbe distributed to only two nodes.

**End-to-End Encryption**

- The encryption process is carried out at the two end systems. The source and the destination share a

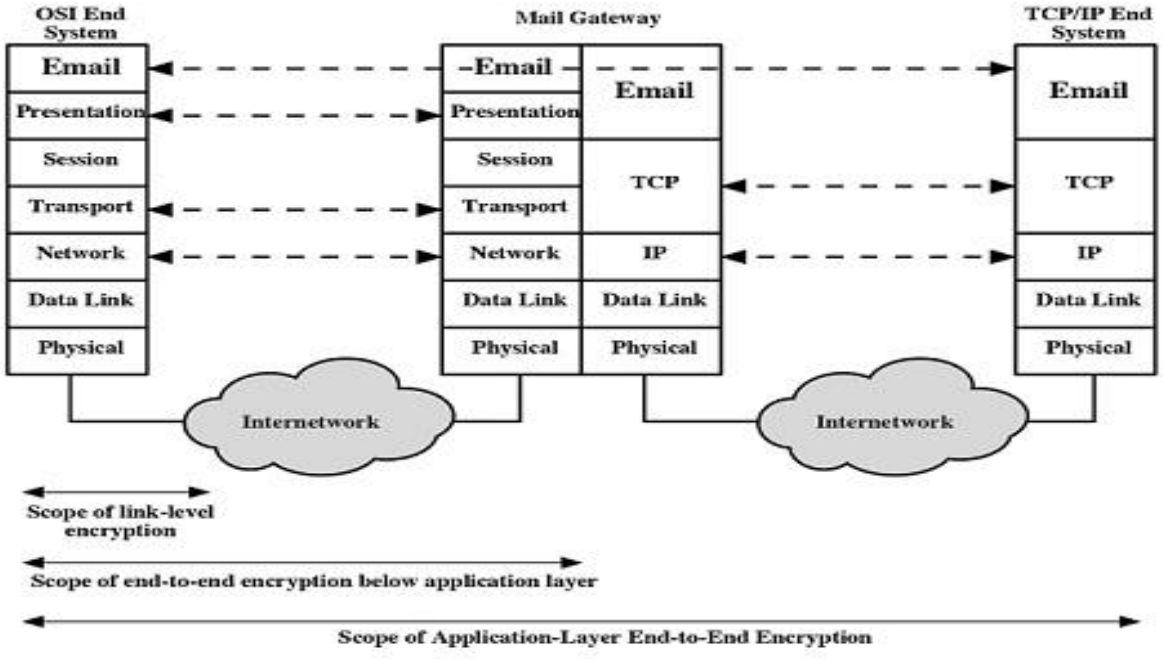
- This plan seems to secure the transmission againstattacks on the network links or switches. There is, however, still a weak spot.
- The source may encrypt only the user data portion, butmust leave the header in the clear.
- With end-to-end encryption, the user data are secure,but the traffic pattern is not. A certain degree of authentication is also provided.

## Link vs. End-to-End Encryptions

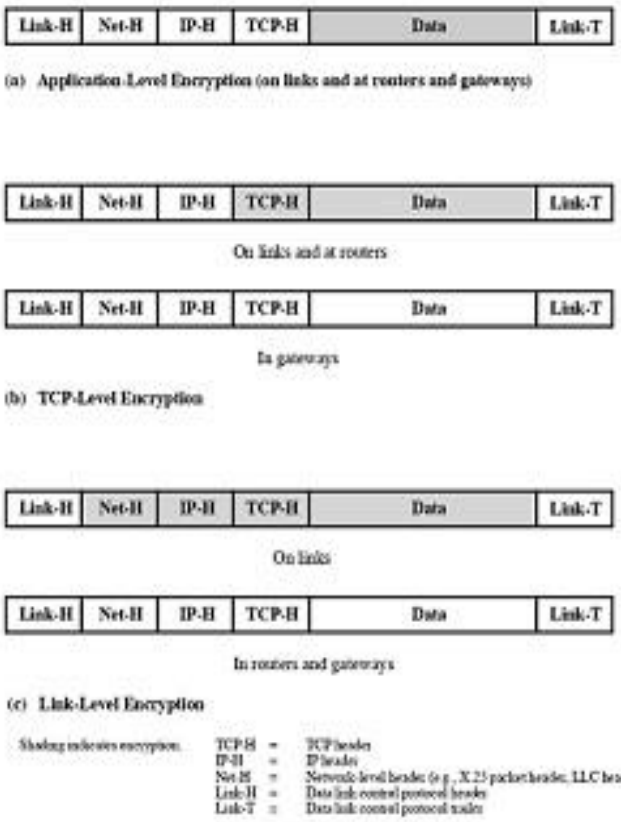
| Link Encryption | End-to-End Encryption |
|---|---|
| *Security within End Systems and Intermediate Systems* | |
| Message exposed in sending host<br>Message exposed in intermediate nodes | Message encrypted in sending host<br>Message encrypted in intermediate nodes |
| *Role of User* | |
| Applied by sending host<br>Transparent to user<br>Host maintains encryption facility<br>One facility for all users<br>Can be done in hardware<br>All or no messages encrypted | Applied by sending process<br>User applies encryption<br>User must determine algorithm<br>Users selects encryption scheme<br>Software implementation<br>User chooses to encrypt, or not, for each message |
| *Implementation Concerns* | |
| Requires one key per (host-intermediate node) pair and (intermediate node-intermediate node) pair<br>Provides host authentication | Requires one key per user pair<br><br>Provides user authentication |

## Deploying End-to-End Encryption

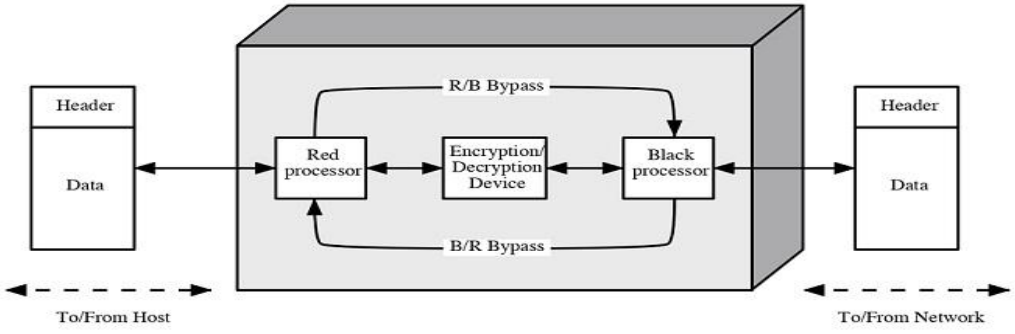
## Store-and-Forward Communications



# Encryption and Protocol Layers




# Front-End Processor Function

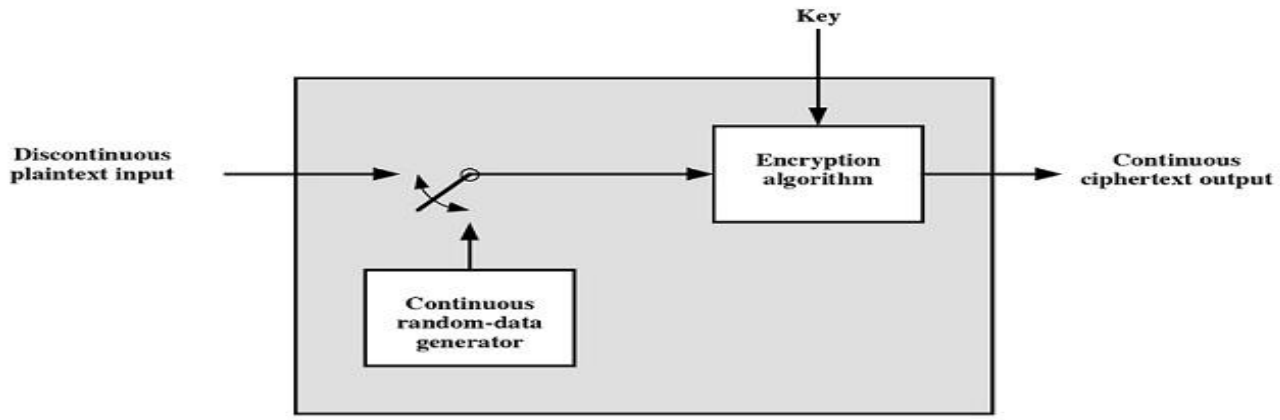


## Traffic Confidentiality

Types of information that can be derived from a traffic analysis attack:

- Identities of partners
- How frequently the partners are communicating
- Message pattern, message length, or quantity of messages
- Events correlated with conversations between particular partners
- Messages of a *covert channel*

## Traffic Padding




## Countering Traffic Analysis

- Link encryption approach
  - packet headers already encrypted
  - further strength via traffic padding
- End-to-end encryption approach: available measures more limited
  - padding out data units to a uniform length
  - inserting null messages randomly

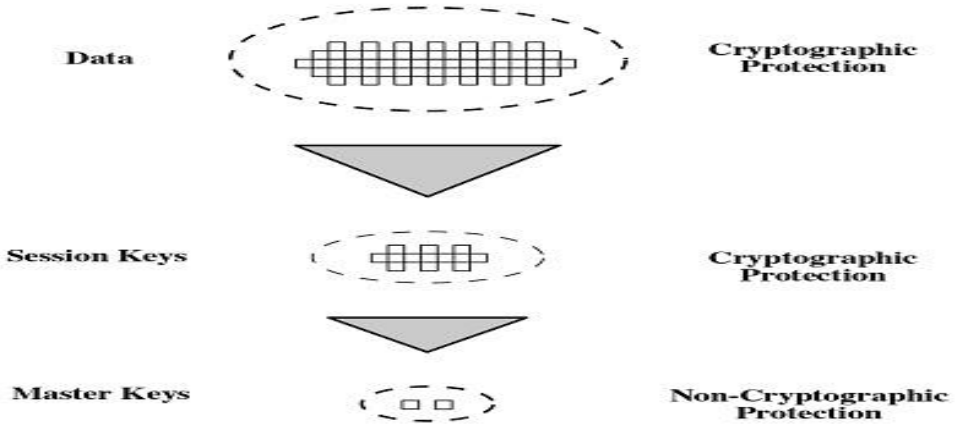
## The Key Distribution Problem

- For symmetric encryption to work, the two parties of exchange must share the same key and that key mus be protected.
- Frequent key changes may be desirable to limit the amount of data compromised.
- The strength of a cryptographic system rests with the technique for solving the key distribution problem—delivering a key to the two parties of an exchange.
- The scale of the problem depends on the number of communication pairs.

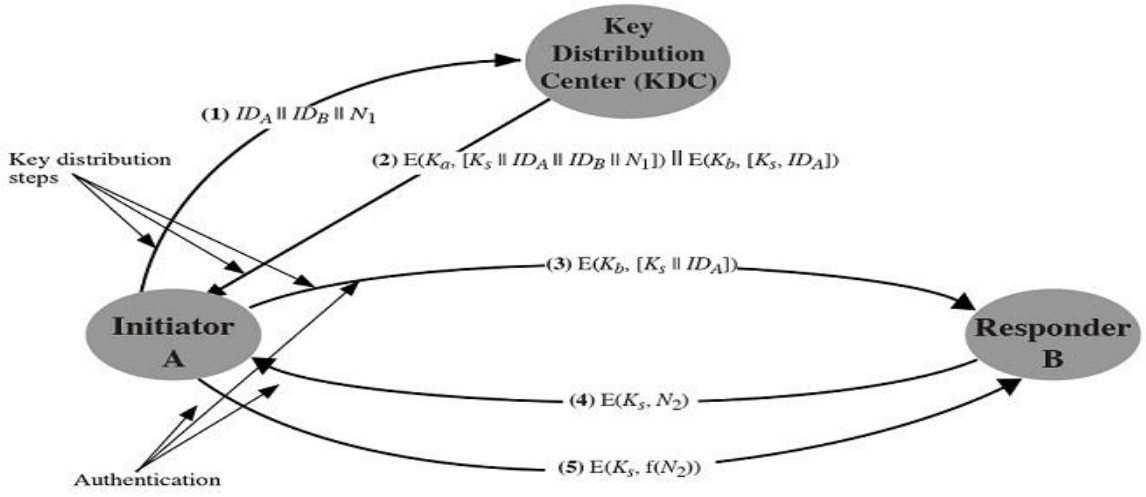
## Using a Key Distribution Center

- A key distribution center is responsible for distribu keys to pairs of users as needed.
- Each user must share a unique key with the key distribution center for purposes of key distribution
- At least two levels of keys must be used: session and master keys.
- If there are $N$ end users, $N(N-1)/2$ session key needed at any one time, but only $N$ master keys required.

## Key Hierarchy



# Key Distribution Scenario




# Hierarchical Key Control

- For large networks, a single KDC is inadequate.
- In a hierarchy of KDCs, each local KDC is responsible for a small domain.
- If the two parties are within the same local domain, the KDC is responsible for key distribution.
- Otherwise, the two corresponding local KDCs can communicate through a global KDC. Any of the three KDCs involved can select the key.
- Advantages: distributing the effort of master key distribution and isolating the damage of a fault.

# Session Key Lifetime

- Two competing considerations in determining the lifetime of a session key:
  - The more frequently session keys are changed, more secure they are.
  - The distribution of session keys delays the start exchange and places a burden on network capa
- The decision can be based on whether the communication protocol is connection-oriented or connectionless.

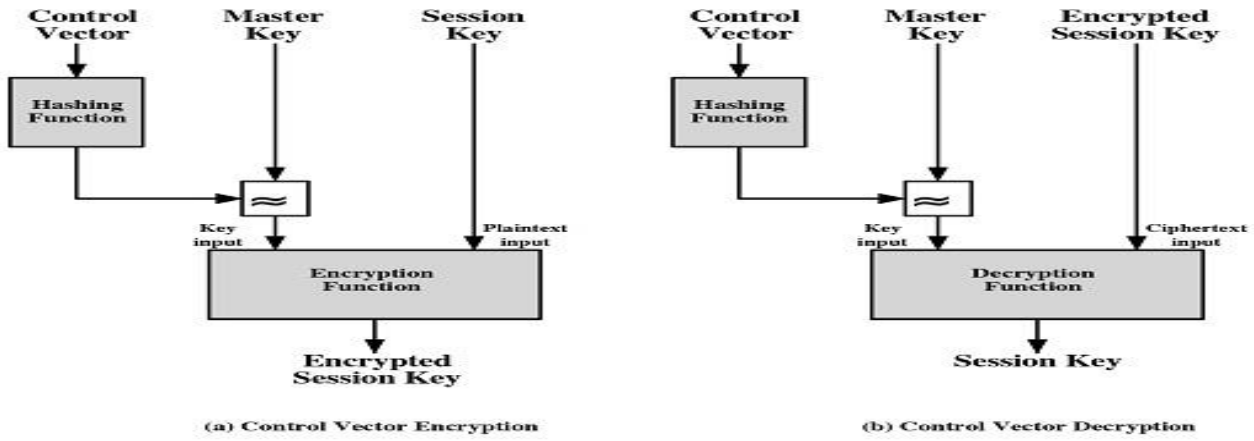
## Decentralized Key Control

- The KDC must be trusted and be protected from subversion.
- This requirement can be avoided if the key distribution is fully decentralized.
- A fully decentralized key control, though not feasible for large networks, may be useful within a local context.
- A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution.

## Controlling Key Usage

- It may be desirable to impose some control on the in which automatically distributed keys are used.
- Possible types of session keys include: data-encry key, PIN-encrypting key, file-encrypting key, etc.
- Key use controlling schemes:
  - Tags
  - Control vectors

## Control Vector



## Pseudorandom Numbers

- True random numbers are hard to come by.
- Cryptographic applications typically use algorithm techniques for random number generation.
- These algorithms are deterministic and therefore produce sequence of numbers that are not statisti random.
- If the algorithm is good, the resulting sequences w pass reasonable tests for randomness.
- Such numbers are often referred to as pseudoran numbers.

## The Use of Random Numbers

- Random numbers are used by a number of secu algorithms for:
  - Nonces (used in authentication protocols)
  - Session key generation (by the KDC or an enc system)
  - Key generation for the RSA algorithm
- Two requirements: randomness and unpredictabi

## Cryptographical Generation

- **Cyclic encryption**: use an arbitrary block ciphe Full-period generating functions are easily obtai
- **DES Output Feedback Mode**: the successive 6 outputs constitute a sequence of pseudorandom numbers.
- **ANSI X9.17 Pseudorandom number generato (PRNG)**: make use of triple DES. Employed in fi security applications and PGP.

## The Linear Congruential Method

| | | |
|---|---|---|
| $m$ | the modulus | $m > 0$ |
| $a$ | the multiplier | $0 \leq a < m$ |
| $c$ | the increment | $0 \leq c < m$ |
| $X_0$ | the starting value (seed) | $0 \leq X_0 < m$ |

- Iterative equation: $X_{n+1} = (aX_n + c) \bmod m$
- Larger values of $m$ imply higher potential for a long period.
- For example, $X_{n+1} = (7^5 X_n) \bmod (2^{31} - 1)$ has a period $2^{31} - 2$.
- What are the weakness and the remedy?

## The Blum Blum Shub (BBS) Generator

- Choose two large prime numbers $p$ and $q$ such that $p \equiv q \equiv 3 \pmod 4$. Let $n = p \times q$.
- Choose a random number $s$ relatively prime to $n$.
- Bit sequence generating algorithm:

$$X_0 = s^2 \bmod n$$
$$\textbf{for } i = 1 \textbf{ to } \infty$$
$$X_i = (X_{i-1})^2 \bmod n$$
$$B_i = X_i \bmod 2$$

- The BBS generator passes the next-bit test.

## Example Operation of BBS Generator

| $i$ | $X_i$ | $B_i$ |
|---|---|---|
| 0 | 20749 | |
| 1 | 143135 | 1 |
| 2 | 177671 | 1 |
| 3 | 97048 | 0 |
| 4 | 89992 | 0 |
| 5 | 174051 | 1 |
| 6 | 80649 | 1 |
| 7 | 45663 | 1 |
| 8 | 69442 | 0 |
| 9 | 186894 | 0 |
| 10 | 177046 | 0 |

| $i$ | $X_i$ | $B_i$ |
|---|---|---|
| 11 | 137922 | 0 |
| 12 | 123175 | 1 |
| 13 | 8630 | 0 |
| 14 | 114386 | 0 |
| 15 | 14863 | 1 |
| 16 | 133015 | 1 |
| 17 | 106065 | 1 |
| 18 | 45870 | 0 |
| 19 | 137171 | 1 |
| 20 | 48060 | 0 |

UNIT-III

## ➢ Public-Key Cryptography

- The development of public-key cryptography is the greatest and perhaps the only true revolution in the entire history of cryptography.
- Earlier all cryptographic systems have been based on the elementary tools of substitution and permutation.
- Public key algorithms are based on **mathematical functions** rather than on substitution and permutation.
- More important, **public-key cryptography is asymmetric**, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key.
- The use of two keys has profound consequences in the areas of confidentiality, key distribution, and authentication, as we shall see.

Before proceeding, we should mention several common misconceptions concerning public-key encryption.

1. Public-key encryption is more secure from cryptanalysis than is symmetric encryption. In fact, the security of any encryption scheme depends on the length of the key and the computational work involved in breaking a cipher.
2. A second misconception is that public-key encryption is a general-purpose technique that has made symmetric encryption obsolete.
3. Finally, there is a feeling that key distribution is important when using public-key encryption, compared to the rather handshaking involved with key distribution centers for symmetric encryption.

Will now discuss the radically different **public key** systems, in which **two keys** are used. Anyone knowing the public key can encrypt messages or verify signatures, but **cannot** decrypt messages or create signatures. The use of two keys has profound consequences in the areas of confidentiality, key distribution, and authentication.

**Terminology Related to Asymmetric EncryptionAsymmetric Keys**

Two related keys, a public key and a private key, that are used to perform complementary operations, such as encryption and decryption or signature generation andsignature verification.

**Public Key Certificate**

A digital document issued and digitally signed by the private key of a Certification Authority that binds the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key.

**Public Key (Asymmetric) Cryptographic Algorithm**

A cryptographic algorithm that uses two related keys, a public key and a private key. The two keys have the property that deriving the private key from the public key is computationally infeasible.

**Public Key Infrastructure (PKI)**

A set of policies, processes, server platforms, software and workstations used for the purpose of administering certificates and public-private key pairs, including the ability to issue, maintain, and revoke public key certificates.

**Principles of Public-Key Cryptosystems**

The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption. The first problem is that of **key distribution.**

As we have seen, key distribution under symmetric encryption requires either

(1) Two communicants already share a key, which somehow has been distributed to them

or (2) The use of a key distribution center.

Whitfield Diffie, one of the discoverers of public-key encryption reasoned that this second requirement negated the very essence of cryptography: the ability to maintain to**tal secrecy** over your own communication.

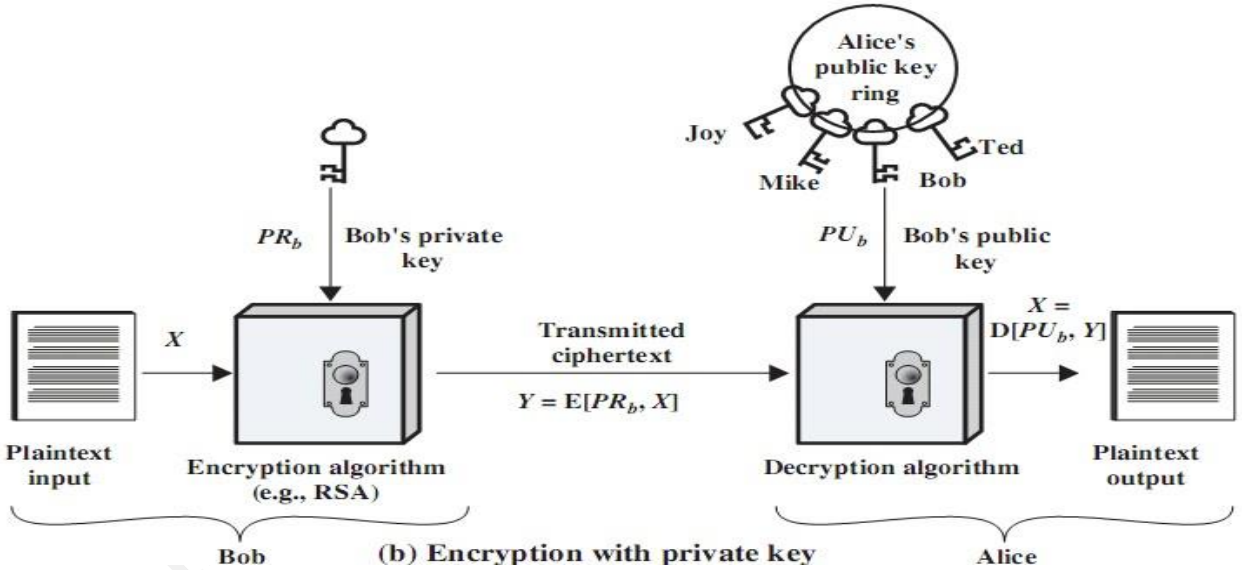
The second problem is **digital signatures.** The electronic messages and documents would need the equivalent of signatures used in paper documents. That is digital message had been sent by a particular person? This is a somewhat broader requirement than that of authentication,

Diffie and Hellman achieved breakthrough by coming up with a method that addressed both problems and was radically different from all previous approaches tocryptography.

**Public-Key Cryptosystems**

Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristic:

- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.



(b) Encryption with private key

- Either of the two related keys can be used for encryption, with the other used fordecryption.

Anyone knowing the public key can encrypt messages or verify signatures, but **cannot** decrypt messages or create signatures, thanks to some clever use of number theory.

A public-key encryption scheme has six ingredients

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

**Ciphertext:** This is the scrambled message produced as output. It depends on theplaintext and the key. For a given message, two different keys will produce two different ciphertexts.

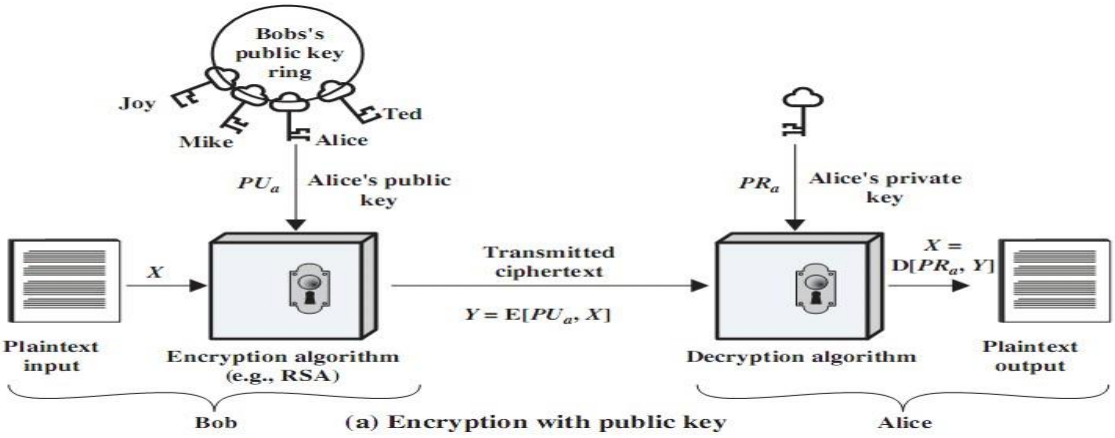
- **Decryption algorithm:** This algorithm accepts the ciphertext and the matching key and produces the original plaintext.

The essential steps are the following:

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

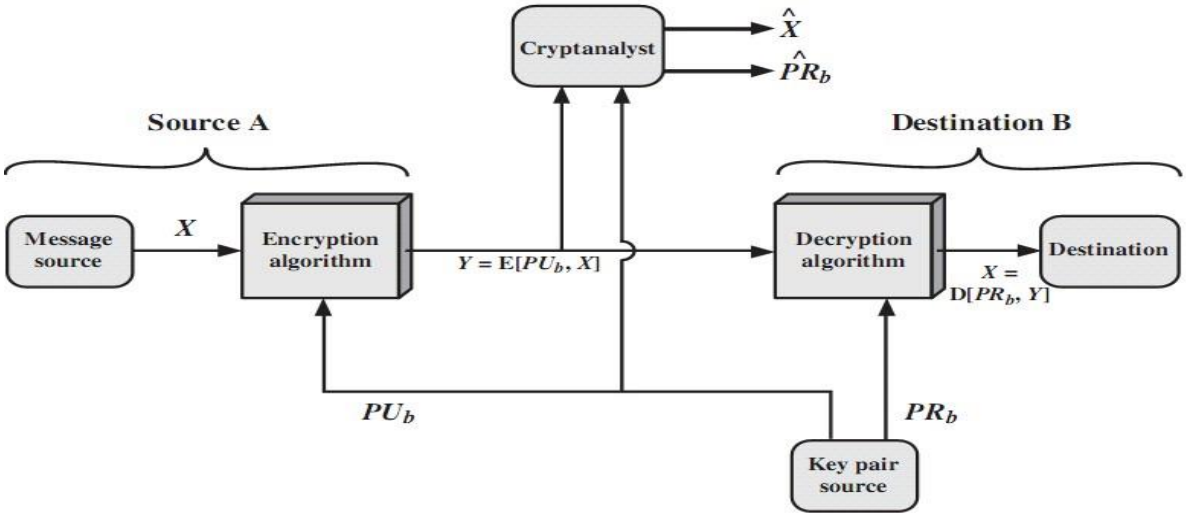
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 9.1a suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

To discriminate between symmetric and public-key encryption, we refer to the key used in symmetric encryption as a **secret key**. The two keys used for asymmetric encryption are referred to as the **public key** and the **private key**. Invariably, the private key is kept secret, but it is referred to as a private key rather than a secret key to avoid confusion with symmetric encryption.




| **Conventional Encryption** | **Public-Key Encryption** |
|---|---|
| Needed to Work: | Needed to Work: |
| 1. The same algorithm with the same key is used for encryption and decryption.<br>2.The sender and receiver must share thealgorithm and the key. | 1. One algorithm is used for encryption anddecryption with a pair of keys, one for encryption and one for decryption.<br>2. The sender and receiver must each have one ofthe matched pair of keys (not the same one). |
| Needed for Security: | Needed for Security: |
| 1. The key must be kept secret.<br>2. It must be impossible or at least<br>3. impractical to decipher a message<br>4. if noother information is available.<br>5. Knowledge of the algorithm plus<br>6. samples of ciphertext must be<br>7. insufficient to determine the key. | One of the two keys must be kept secret.<br>It must be impossible or at least impractical todecipher a message if no other information is available.<br>Knowledge of the algorithm plus one of thekeys plus samples of ciphertext must be insufficient to determine the other key. |

 Public-key schemes can be used for either secrecy or authentication, or both (as shown here). There is some source A that produces a message in plaintext X. The message is intended for




destination B.

- B generates a related pair of keys: a public key, PUb, and a private key, PRb. PRb is known only to B, whereas PUb is publicly available and therefore accessible by A.
- With the message X and the encryption key PUb as input, A forms the ciphertext

  Y = E(PUb, X)
- The intended receiver, in possession of the matching private key, is able to invert the transformation: X = D(PRb, Y).
- An adversary, observing Y and having access to PUb, but not having access to PRb or X, must attempt to recover X and/or PRb. This provides confidentiality.
- Public-key encryption can also provide authentication:

  Y = E (PRa, X); X = D(PUa, Y)
- To provide both the authentication function and confidentiality have a double use of the public-key scheme (as shown here):

  Z = E (PUb, E (PRa, X))

  X = D (PUa, D (PRb, Z))

**Applications for Public-Key Cryptosystems**

We can classify the use of public-key cryptosystems into three categories:

- **Encryption/decryption:** The sender encrypts a message with the recipient'spublic key.
- **Digital signature:** The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.
- **Key exchange:** Two sides cooperate to exchange a session key. Several different approaches are possible, involving the private key(s) of one or both parties.

**Requirements for Public-Key Cryptography**

1. It is computationally easy for a party B to generate a pair (public key $PU_b$, privatekey $PR_b$).
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:

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

A **one-way function** is one that maps a domain into a range such that every function value has a unique inverse, with the condition that the calculation of the function is easy whereas the calculation of the inverse is infeasible:

$$Y = f(X) \text{ easy}$$
$$X = f^{-1}(Y) \text{ infeasible}$$

**Trap-door one-way function**

It is easy to calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known.

$$Y = f_k(X) \text{ easy, if k and X are known } X = f^{-1}(Y) \text{ easy, if k and Y are known}$$
$$X = f_k^{-1}(Y) \text{ infeasible, if Y known but k not known}$$

**Public-Key Cryptanalysis**

- As with symmetric encryption, a public-key encryption scheme is vulnerable to a **brute-force attack.** The countermeasure is the same: **Use large keys.**
- The key size must be large enough to make **brute-force attack impractical** but result in encryption/decryption speeds that are too slow for general-purpose use.
- Another form of attack is to find some way to **compute the private key** given the public key. To date, it has not been mathematically proven that this form of attack is infeasible for a particular public-key algorithm.
- Finally, there is a form of attack that is peculiar to public-key systems. This is, in essence, a **probable-message attack**.
- Suppose, for example, that a message were to be sent that consisted solely of a 56-bit DES key. An adversary could encrypt all possible 56-bit DES keys using the public key and could discover the encrypted key by matching the transmitted ciphertext.
- Thus, no matter how large the key size of the public-key scheme, the attack is reduced to a brute-force attack on a 56-bit key. This attack can be thwarted by **appending some random bits** to such simple messages.

## ➢ Introduction to Number Theory:

Number Theory plays an important role in encryption algorithm. Cryptography is the practice of hiding information, converting some secret information to not readable texts.

**INTRODUCTION**

For thousands of years people have searched for the way to send a message secretly. There is a story that, in ancient time, a king needed to send a secret message to his general in battle. The king took a servant, shaved his head and wrote the message on his head. He waited for the servant's hair to grow back and then sent the servant to the general. The general then shaved the servant's head and read the message. If the enemy had captured the servant, they presumably would not have known to shave his head and message would have been safe.

Cryptography is the study of methods to send and receive the secret messages. In general we have a

sender who is trying to send a message to receiver. There is also an adversary, who wants to steal the message. We are successful if sender is able to communicate a message to the receiver without adversary learning what the message was.

We will use some important concepts of Number Theory and Cryptography which are given below:

**Important concepts in Number Theory**

**Prime Numbers-** A positive integer $p$ is said to be a prime if it has only two factors namely 1 and $p$ itself. For Example: Primes are 2, 3, 5, 7, 11, 13, 17 …

**Divisors:** A positive integer a is said to divide an integer b if there exist an integer c such that b = a.c and written as $a \mid b$.

for example $2 \mid 10$ as 10 = 2.5 but 3 do not divide 10 as there does not exist any integer $c$ such that 10 = 3. $c$

**Greatest Common Divisor:** Let a and b be two positive integers then an integer d is called greatest common divisor of a and b if if d/a and d is common divisor of a and b.

And if any integer c is such that c | a and c/b then c/d i.e any other common divisor of a and b will divide it denoted by d= (a, b). For Example: 6 = (24,30) Two numbers a and b are said to relatively prime or co prime if their greatest common divisor is 1 i.e. $(a, b) = 1$

For Example: 10 and 11 are co prime

Congruence: Let a and b be two integers and m is any positive integer then a is said to congruent to b modulo m if m divide difference of a and b i.e for m/a-b.it is denoted by $a \equiv b \pmod{m}$

## ➢ Prime Numbers

A central concern of number theory is the study of prime numbers. An integer p > 1 is a prime number if and only if its only divisors are $\pm 1$ and $\pm p$ Prime numbers play a critical role in number theory.

Any integer a >1 can be factored in a unique way as $a = p1^{a1} * p2^{a2} * \ldots\ldots\ldots * pt^{at.}$

where p1 < p2 <……. < pt are prime numbers and where each ai is a positive integer. This is known as the fundamental theorem of arithmetic; a proof can be found in any text on number theory.

$91 = 13*7$

$3600 = 2^4 * 3^2 * 5^2$

$11011 = 7^8 * 11^2 * 13$

It is useful for what follows to express another way. If P is the set of all prime numbers, then any positive integer a can be written uniquely in the following form:

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

The right-hand side is the product over all possible prime numbers p; for any particular value of a, most of the exponents ap will be 0. The value of any given positive integer can be specified by simply listing all the nonzero exponents in the foregoing formulation.

## ➢ Modular Arithmetic (Clock Arithmetic)

Modular arithmetic is a system of arithmetic for integers, where values reset to zero and begin to increase again, after reaching a certain predefined value, called the modulus (modulo). Modular arithmetic is widely used in computer science and cryptography.

**Definition** Let $Z_N$ be a set of all non-negative integers that are smaller than N:

$Z_N = \{0, 1, 2, ..., N-1\}$

where:

- **N** is a positive integer,
- if **N** is a prime, it will be denoted **p** (and the whole set as $\mathbf{Z_p}$).

To determine the value of an integer for a modulus N, one should divide this number by N. Its value in $Z_N$ is equal to the remainder of the division. In modular arithmetic, it is possible to define all typical operations, as in normal arithmetic. They work as one may expect. It is possible to use the same commutative, associative, and distributive laws.

**Modular inversion**

Integers in modular arithmetic may (but not must) have inverse numbers.

**Definition** The inverse of x in $Z_N$ is a number y in $Z_N$, that satisfies the equation:

$x \cdot y = 1$ (in $Z_N$)

where:

- **y** is denoted $\mathbf{x^{-1}}$

For example, if N is an odd number, then the inverse of 2 in $Z_N$ is (N+1)/2:

$x \cdot (N+1)/2 = N + 1 = 1$ (in $Z_N$)

**Theorem** A number x is invertible in $Z_N$ if and only if the numbers x and N are relatively prime.

- The theorem can be proved using the fact that it is possible to present the greatest common divisor of two integers as a sum of two products each of the numbers and another properly selected integer:

  $a \cdot x + b \cdot y = \gcd(x,y)$

Determining inverse numbers in $Z_N$ allows solving linear equations in modular arithmetic:

the equation: $a \cdot x + b = 0$ (in $Z_N$)

has the solution: $x = -b \cdot a^{-1}$ (in $Z_N$)

**Definition** The symbol $Z^*_N$ denotes a set of all elements of $Z_N$ that are invertible in $Z_N$; that means the set of numbers x that belong to $Z_N$, and x and N are relatively prime.

for example for a prime number p:

$Z^*p = Zp \setminus \{0\} = \{1, 2, \ldots, p - 1\}$

## ➢ Euler's Theorem

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

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

**Proof:** Equation (8.4) is true if n is prime, because in that case, $\phi(n) = (n - 1)$ and Fermat's theorem holds. However, it also holds for any integer n. Recall that f(n) is the number of positive integers less than n that are relatively prime to n. Consider the set of such integers, labeled as

$R = \{x1, x2, \ldots\ldots, x\phi(n)\}$

That is, each element xi is a unique positive integer less than n with $\gcd(xi, n) = 1$. Now multiply each element by a, modulo n:

$S = \{(ax1 \bmod n), (ax2 \bmod n), \ldots\ldots, (ax\phi(n) \bmod n)\}$

The set S is a permutation6 of R, by the following line of reasoning:

**1.** Because a is relatively prime to n and xi is relatively prime to n, axi must also be relatively prime to n. Thus, all the members of S are integers that are less than n and that are relatively prime to n.

1. There are no duplicates in S. Refer to Equation (4.5). If $axi \bmod n = axj \bmod n$, then xi = xj.

Therefore

$$\prod_{i=1}^{\phi(n)} (ax_i \bmod n) = \prod_{i=1}^{\phi(n)} x_i$$

$$\prod_{i=1}^{\phi(n)} ax_i \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

$$a^{\phi(n)} \times \left[\prod_{i=1}^{\phi(n)} x_i\right] \equiv \prod_{i=1}^{\phi(n)} x_i \pmod{n}$$

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

which completes the proof. This is the same line of reasoning applied to the proof of Fermat's theorem.

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

As is the case for Fermat's theorem, an alternative form of the theorem is also useful:

$$a^{\phi(n)+1} \equiv a \pmod{n}$$

## ➢ Primary and Factorization

**Applications of Prime Factorization**

Prime factorization is used extensively in the real world. The two most important applications of prime factorization are given below.

- Cryptography and Prime Factorization
- HCF and LCM Using Prime Factorization

**Cryptography and Prime Factorization**

Cryptography is a method of protecting information using codes. Prime factorization plays an important role for the coders who create a unique code using numbers which is not too heavy for computers to store or process quickly.

In information security, large semi-primes have applications in encryption algorithms. They are used for generating public keys and private keys, such as in the Rivest–Shamir– Adleman (RSA) cryptosystems . The property that the prime factorization of large numbers is a challengingly difficult task is well utilized in RSA-based encryption algorithms. Due to the dominant application of the RSA public-key primitive in cryptography, the security of RSA has been extensively analyzed for various attack scenarios.

**Special-purpose**

A special-purpose factoring algorithm's running time depends on the properties of the number to be factored or on one of its unknown factors: size, special form, etc. The parameters which determine the running time vary among algorithms.

An important subclass of special-purpose factoring algorithms is the Category 1 or First Category algorithms, whose running time depends on the size of smallest prime factor. Given an integer of unknown form, these methods are usually applied before general-purpose methods to remove small factors.[9] For example, naive trial division is a Category 1 algorithm.

- Trial division
- Wheel factorization
- Pollard's rho algorithm
- Algebraic-group factorisation algorithms, among which are Pollard's p − 1 algorithm, Williams' p + 1 algorithm, and Lenstra elliptic curve factorization
- Fermat's factorization method
- Euler's factorization method
- Special number field sieve

## ➢ DISCRETE LOGARITHMS

Discrete logarithms are fundamental to a number of public-key algorithms, including Diffie-Hellman key exchange and the digital signature algorithm (DSA).

**The Powers of an Integer, Modulo n**

Recall from Euler's theorem that, for every a and n that are relatively prime,

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

where ϕ(n), Euler's totient function, is the number of positive integers less than n and relatively prime to n. Now consider the more general expression:

$$a^{m} \equiv 1 \pmod{n}$$

If a and n are relatively prime, then there is at least one integer m that satisfies Equation namely, M = ϕ(n). The least positive exponent m for which Equation (8.10) holds is referred to in several ways:

- The order of a (mod n)
- The exponent to which a belongs (mod n)
- The length of the period generated by a

To see this last point, consider the powers of 7, modulo 19:

$$\begin{aligned}
7^1 &\equiv & &7 \pmod{19}\\
7^2 &= 49 = 2 \times 19 + 11 &\equiv\ &11 \pmod{19}\\
7^3 &= 343 = 18 \times 19 + 1 &\equiv\ &1 \pmod{19}\\
7^4 &= 2401 = 126 \times 19 + 7 &\equiv\ &7 \pmod{19}\\
7^5 &= 16807 = 884 \times 19 + 11 &\equiv\ &11 \pmod{19}
\end{aligned}$$

There is no point in continuing because the sequence is repeating. This can be proven by noting that $7^3 \equiv 1 \pmod{19}$, and therefore, $7^{3+j} \equiv 7^3 7^j \equiv 7^j \pmod{19}$, and hence, any two powers of 7 whose exponents differ by 3 (or a multiple of 3) are congruent to each other (mod 19). In other words, the sequence is periodic, and the length of the period is the smallest positive exponent $m$ such that $7^m \equiv 1 \pmod{19}$.

Table 8.3 shows all the powers of a, modulo 19 for all positive a 6 19. The length of the sequence for each base value is indicated by shading. Note the following:

**1.** All sequences end in 1. This is consistent with the reasoning of the preceding few paragraphs.

**2.** The length of a sequence divides $\phi(19) = 18$. That is, an integral number of sequences occur i n each row of the table.

**3.** Some of the sequences are of length 18. In this case, it is said that the base integer a generates (via powers) the set of nonzero integers modulo 19. Each such integer is called a primitive root of the modulus 19.

Table 8.3 Powers of Integers, Modulo 19

| $a$ | $a^2$ | $a^3$ | $a^4$ | $a^5$ | $a^6$ | $a^7$ | $a^8$ | $a^9$ | $a^{10}$ | $a^{11}$ | $a^{12}$ | $a^{13}$ | $a^{14}$ | $a^{15}$ | $a^{16}$ | $a^{17}$ | $a^{18}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 4 | 8 | 16 | 13 | 7 | 14 | 9 | 18 | 17 | 15 | 11 | 3 | 6 | 12 | 5 | 10 | 1 |
| 3 | 9 | 8 | 5 | 15 | 7 | 2 | 6 | 18 | 16 | 10 | 11 | 14 | 4 | 12 | 17 | 13 | 1 |
| 4 | 16 | 7 | 9 | 17 | 11 | 6 | 5 | 1 | 4 | 16 | 7 | 9 | 17 | 11 | 6 | 5 | 1 |
| 5 | 6 | 11 | 17 | 9 | 7 | 16 | 4 | 1 | 5 | 6 | 11 | 17 | 9 | 7 | 16 | 4 | 1 |
| 6 | 17 | 7 | 4 | 5 | 11 | 9 | 16 | 1 | 6 | 17 | 7 | 4 | 5 | 11 | 9 | 16 | 1 |
| 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 | 7 | 11 | 1 |
| 8 | 7 | 18 | 11 | 12 | 1 | 8 | 7 | 18 | 11 | 12 | 1 | 8 | 7 | 18 | 11 | 12 | 1 |
| 9 | 5 | 7 | 6 | 16 | 11 | 4 | 17 | 1 | 9 | 5 | 7 | 6 | 16 | 11 | 4 | 17 | 1 |
| 10 | 5 | 12 | 6 | 3 | 11 | 15 | 17 | 18 | 9 | 14 | 7 | 13 | 16 | 8 | 4 | 2 | 1 |
| 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 | 11 | 7 | 1 |
| 12 | 11 | 18 | 7 | 8 | 1 | 12 | 11 | 18 | 7 | 8 | 1 | 12 | 11 | 18 | 7 | 8 | 1 |
| 13 | 17 | 12 | 4 | 14 | 11 | 10 | 16 | 18 | 6 | 2 | 7 | 15 | 5 | 8 | 9 | 3 | 1 |
| 14 | 6 | 8 | 17 | 10 | 7 | 3 | 4 | 18 | 5 | 13 | 11 | 2 | 9 | 12 | 16 | 15 | 1 |
| 15 | 16 | 12 | 9 | 2 | 11 | 13 | 5 | 18 | 4 | 3 | 7 | 10 | 17 | 8 | 6 | 14 | 1 |
| 16 | 9 | 11 | 5 | 4 | 7 | 17 | 6 | 1 | 16 | 9 | 11 | 5 | 4 | 7 | 17 | 6 | 1 |
| 17 | 4 | 11 | 16 | 6 | 7 | 5 | 9 | 1 | 17 | 4 | 11 | 16 | 6 | 7 | 5 | 9 | 1 |
| 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 | 18 | 1 |

More generally, we can say that the highest possible exponent to which a number can belong (mod n) is $\phi(n)$. If a number is of this order, it is referred to as a **primitive root** of n. The importance of this not ion is that if a is a primitive root of n, then its powers

$$a, a^2, \ldots, a^{\phi(n)}$$

are distinct (mod n) and are all relatively prime to n. In particular, for a prime number p, if a is a primitiv e root of p, then

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

are distinct (mod p). For the prime number 19, its primitive roots are 2, 3, 10, 13, 14, and 15.
Not all integers have primitive roots. In fact, the only integers with primitive roots are those of the form 2, 4, pa, and 2pa, where p is any odd prime and a is a positive integer.

**Logarithms for Modular Arithmetic**

With ordinary positive real numbers, the logarithm function is the inverse of exponentiation. An analogous function exists for modular arithmetic.
Let us briefly review the properties of ordinary logarithms. The logarithm of a number is defined to be the power to which some positive base (except 1) must be
raised in order to equal the number. That is, for base x and for a value y,

$$y = x^{\log_x(y)}$$

The properties of logarithms include

$$\log_x(1) = 0$$
$$\log_x(x) = 1$$
$$\log_x(yz) = log_x(y) + \log_x(z)$$
$$\log_x(y^r) = r \times \log_x(y)$$

Consider a primitive root a for some prime number p (the argument can be developed for non primes as well). Then we know that the powers of a from 1 through (p - 1) produce each integer from 1 through (p - 1) exactly once. We also know that any integer b satisfies

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

by the definition of modular arithmetic. It follows that for any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$$b \equiv r \pmod{p} \quad \text{for some } r, \text{ where } 0 \le r \le (p-1)$$

This exponent i is referred to as the **discrete logarithm** of the number b for the base a (mod p). We denote this value as dloga,p(b).10

Note the following:

dloga,p(1) = 0 because a0 mod p = 1 mod p = 1

dloga,p(a) = 1 because a1 mod p = a

Here is an example using a nonprime modulus, $n = 9$. Here $\phi(n) = 6$ and $a = 2$ is a primitive root. We compute the various powers of $a$ and find

$2^0 = 1 \quad 2^4 = 7 \pmod 9$

$2^1 = 2 \quad 2^5 = 5 \pmod 9$

$2^2 = 4 \quad 2^6 = 1 \pmod 9$

$2^3 = 8$

This gives us the following table of the numbers with given discrete logarithms (mod 9) for the root $a = 2$:

| Logarithm | 0 | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|---|
| Number | 1 | 2 | 4 | 8 | 7 | 5 |

To make it easy to obtain the discrete logarithms of a given number, we rearrange the table:

| Number | 1 | 2 | 4 | 5 | 7 | 8 |
|---|---|---|---|---|---|---|
| Logarithm | 0 | 1 | 2 | 5 | 4 | 3 |

Now consider

$$x = a^{\text{dlog}_{a,p}(x)} \bmod p \qquad y = a^{\text{dlog}_{a,p}(y)} \bmod p$$

$$xy = a^{\text{dlog}_{a,p}(xy)} \bmod p$$

Using the rules of modular multiplication,

$$xy \bmod p = [(x \bmod p)(y \bmod p)] \bmod p$$

$$a^{\text{dlog}_{a,p}(xy)} \bmod p = \left[\left(a^{\text{dlog}_{a,p}(x)} \bmod p\right)\left(a^{\text{dlog}_{a,p}(y)} \bmod p\right)\right] \bmod p$$

$$= \left(a^{\text{dlog}_{a,p}(x)+\text{dlog}_{a,p}(y)}\right) \bmod p$$

But now consider Euler's theorem, which states that, for every $a$ and $n$ that are relatively prime,

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

Any positive integer $z$ can be expressed in the form $z = q + k\phi(n)$, with $0 \leq q < \phi(n)$. Therefore, by Euler's theorem,

$$a^z \equiv a^q (\text{mod } n) \qquad \text{if } z \equiv q \bmod \phi(n)$$

Applying this to the foregoing equality, we have

$$\text{dlog}_{a,p}(xy) \equiv [\text{dlog}_{a,p}(x) + \text{dlog}_{a,p}(y)](\text{mod}\phi(p))$$

and generalizing,

$$\text{dlog}_{a,p}(y^r) \equiv [r \times \text{dlog}_{a,p}(y)](\text{mod } \phi(p))$$

This demonstrates the analogy between true logarithms and discrete logarithms.
Keep in mind that unique discrete logarithms mod m to some base a exist only if a is a primitive root of m.
Table 8.4, which is directly derived from Table 8.3, shows the sets of discrete logarithms that can be defined for modulus 19.

**Table 8.4** Tables of Discrete Logarithms, Modulo 19

Table 8.4 Tables of Discrete Logarithms, Modulo 19

**(a) Discrete logarithms to the base 2, modulo 19**

| $a$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $log_{2,19}(a)$ | 18 | 1 | 13 | 2 | 16 | 14 | 6 | 3 | 8 | 17 | 12 | 15 | 5 | 7 | 11 | 4 | 10 | 9 |

**(b) Discrete logarithms to the base 3, modulo 19**

| $a$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $log_{3,19}(a)$ | 18 | 7 | 1 | 14 | 4 | 8 | 6 | 3 | 2 | 11 | 12 | 15 | 17 | 13 | 5 | 10 | 16 | 9 |

**(c) Discrete logarithms to the base 10, modulo 19**

| $a$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $log_{10,19}(a)$ | 18 | 17 | 5 | 16 | 2 | 4 | 12 | 15 | 10 | 1 | 6 | 3 | 13 | 11 | 7 | 14 | 8 | 9 |

**(d) Discrete logarithms to the base 13, modulo 19**

| $a$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $log_{13,19}(a)$ | 18 | 11 | 17 | 4 | 14 | 10 | 12 | 15 | 16 | 7 | 6 | 3 | 1 | 5 | 13 | 8 | 2 | 9 |

**(e) Discrete logarithms to the base 14, modulo 19**

| $a$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $log_{14,19}(a)$ | 18 | 13 | 7 | 8 | 10 | 2 | 6 | 3 | 14 | 5 | 12 | 15 | 11 | 1 | 17 | 16 | 4 | 9 |

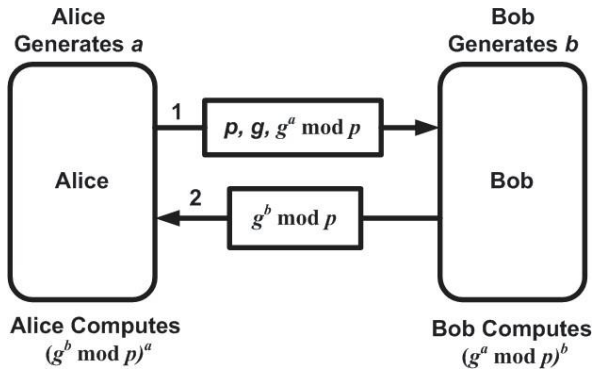
**(f) Discrete logarithms to the base 15, modulo 19**

| $a$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $log_{15,19}(a)$ | 18 | 5 | 11 | 10 | 8 | 16 | 12 | 15 | 4 | 13 | 6 | 3 | 7 | 17 | 1 | 2 | 14 | 9 |

## ➢ Diffie Hellman Algorithm

Diffie Hellman (DH) key exchange algorithm is a method for securely exchanging cryptographic keys over a public communications channel. Keys are not actually exchanged – they are jointly derived. It is named after their inventors Whitfield Diffie and Martin Hellman.
If Alice and Bob wish to communicate with each other, they first agree between them a large prime number p, and a generator (or base) g (where $0 < g < p$).

Alice chooses a secret integer a (her private key) and then calculates g^a mod p (which is her public key). Bob chooses his private key b, and calculates his public key in the same way.
Bob knows b and g^a, so he can calculate (g^a)^b mod p = g^ab mod p. Therefore both Alice and Bob know a shared secret g^ab mod p. An eavesdropper Eve who was listening in on the communication knows p, g, Alice's public key (g^a mod p) and Bob's public key (g^b mod p). She is unable to calculate the shared secret from these values.
In static-static mode, both Alice and Bob retain their private/public keys over multiple communications. Therefore the resulting shared secret will be the same every time. In ephemeral-static mode one party will generate a new private/public key every time, thus a new shared secret will be generated.
Diffie–Hellman Concept The Diffie–Hellman protocol is based on the discrete logarithm problem (DLP). The protocol requires two large numbers p and g, where, p and g are both publicly available numbers. Parameter p is a prime number with at least 512 bits and parameter g (called a generator) is an integer less than p, with the property: for every number n between 1 and p-1 inclusive, there is a power k of g such that $n = g^k \bmod p$. Suppose Alice and Bob want to agree on a shared secret key using the Diffie–Hellman key agreement protocol, they will generate shared key as follows: first, Alice generates a random private value a and Bob generates a random private value b. Both a and b are drawn from the same set of integers. Then they derive their public values using parameters p and g and their private values. Alice's public value is $x = g^a \bmod p$ and Bob's public value is $y = g^b \bmod p$. They then exchange public values x and y. Finally, Alice computes $ka = y^a \bmod p$, and Bob computes $kb = x^b \bmod p$. Since ka = kb = k, Alice and Bob now have a shared secret key k . Figure 1 depicts a simplified schematic of the Diffie–Hellman key exchange.




**A. Diffie–Hellman Key Exchange Algorithm**

Let p be a large prime and assume that α is a primitive element of $Z_p$, p and α are publicly known [4].

1. Alice choose a $(0 \le a \le -2)$at random.
2. Alice computes $x = g^a \bmod p$ and sends it to Bob.
3. Bob chooses b $(0 \le b \le -2)$at random.
4. Bob computes $y = g^b \bmod p$ and sends it to Alice.
5. Alice computes $(g^b)^a \bmod p$ whereas Bob computes $(g^a)^b \bmod p$. In other words, both Alice and Bob compute the same key $g^{ab} \bmod p$. Figure 2 depicts an example of the Diffie–Hellman key exchange. If p = 17014118346046923173168730371 5884105727, then it would take roughly $1.14824 \times 10^{21}$

steps to solve, and each step requires many calculations. Even using Google′ s computers which are estimated to perform 300 trillion calculations per second, it would take roughly 5 years to solve

**B.** Security of Diffie–Hellman If eavesdropper,

**C.** Mallory, learns integers p, g, x, y but not the discrete logarithm a of x and b of y to the base g. She wants to determine the secret key $k = g^{ab} \bmod p$ from p, g, x, y. This is called DiffieHellman problem. She can compute discrete logarithms mod p, and try to solve the Diffie-Hellman problem. She determines the discrete logarithm b of y to the base g and computes the key $k = x^b$.This is the only known method for breaking the DiffieHellman protocol. Until now, no one has succeeded in breaking Diffie-Hellman problem. It is an

important open problem of public-key cryptography to find such a proof. As long as the Diffie-Hellman problem is difficult to solve, no eavesdropper can determine the secret key from publicly known information

| Common Number = 2 | |
|---|---|
| Random number = 4 | Random number = 7 |
| $2^4 = 16$ | $2^7 = 128$ |
| 128 | 16 |
| $128^4 = 268,435,456$ | $16^7 = 268,435,456$ |

C. Applications of Diffie–Hellman in Network Protocols Diffie-Hellman is currently used in many network protocols, such as: • Secure Sockets Layer (SSL)/Transport Layer Security (TLS). • Secure Shell (SSH). • Internet Protocol Security (IPSec). • Public Key Infrastructure (PKI). • In all major VPN gateway's today (PKI).

## INTRODUCTION

RSA is a public-key algorithm that based on mathematical functions rather than on substitution and permutation operations. RSA was developed in 1977 by Ron Rivest, Adi Shamir, and Len Adleman and first published in 1978. The Rivest-Shamir-Adleman (RSA) scheme has since that time reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption.

The security of RSA was based on the number of bits in private-key .The encryption and decryption in RSA requires taking heavy exponential multiplications modulus of a large integer *N* which is the product of two large primes' p and q .

The encryption and decryption time in RSA are roughly proportional to the number of bits in public and private exponents respectively. To reduce the encryption time, one may wish to use a small public exponent e, and to reduce the decryption time, a short secret exponent d can be used. In this context many RSA variants are designed.

The wide range of applications today raises the issue of what is the adequate security option that satisfies the application's security requirements. For applications applying encryption the question may be tailored to what RSA variant is more appropriate.

## ➢ OVERVIEW OF RSA VARIANTS

This section describes the Standard RSA and the RSA variants that had been designed to decrease the decryption time.

### A. RSA standard

The original RSA cryptosystem was proposed in 1978 by Rivest, Shamir and Adelman and consists of three parts , described in the following subsections.

### 1. RSA Key generation

1- Generate two different primes $p$ and $q$ of $(n/2)$-bit each. 2- Compute $N = pq$ and $\emptyset(N) = (p-1)(q-1)$.Choose a random integer $1 < e < \emptyset(N)$ such that $\gcd(e,\emptyset(N)) = 1$.

3- Next, compute the uniquely defined integer $1 < d < \emptyset(N)$ satisfying $ed \equiv 1 \pmod{\emptyset(N)}$.The public key is $<N, e>$ and the private key $<N, d>$.

### 2. RSA Encryption

To encrypt a message $X$ with the public key $<N, e>$, transform the message $X$ to an integer $M$ in $\{0,\ldots,N-1\}$ and compute the ciphertext $C = M^e \bmod N$.

### 3. RSA Decryption

To decrypt the ciphertext $C$ with the private key $<N, d>$, compute $M = C^d \bmod N$ and employ

the reversetransformation to obtain the message *X* from *M* [3][1].

### 4. Complexity of RSA algorithm

The calculate the complexity of RSA algorithm, the exponentiations of the form $C^d \ mod\ N$ take time $O(\ dM(n))$,where $M(n)$ is the cost of multiplying two n-bit integers and can be taken as $O(n^2\ )$. And $|d| \approx |n|$, recall that the number of binary operations to compute $C^d mod\ N$ is $1.5n.n^2 = 1.5n^3$, thus these algorithms take time $O(n^3)$.

On another hand $Me\ (mod\ N)$ take time $O\ (eM\ (n))$, |e| is small, thus the encryption algorithms take time

$O(n^2)$. This means that encryption is much faster than decryption in RSA Standard.

## B. Standard RSA with CRT

Based on the Chinese Remainder Theorem (CRTproposed an RSA variant,RSA-CRT, to speed up RSA decryption.

Here we review the basic steps constituting the RSA, together with the frequently used technique CRT. Suppose the message is M, the two large primes are p and q, the modulus is $N = pq$, and public key is $<N, e>$, private key is $<N, d>$ and to decrease decryption time private key is $<dp, dq, p, q>$ such that $dp \equiv d\ mod (p-1)$ and $dq \equiv d\ mod\ (q-1)$.

The encryption/decryption steps are as follows:

### 1. RSA-CRT Encryption

***E (M)***: $C = M^e\ mod\ N$

### 2. RSA-CRT Decryption

Instead of directly computing $M = C^d\ mod\ N$, the decryption algorithm evaluates $Mp = C^{dp}\ mod\ p$ and $Mq = C^{dq}\ mod\ q$ where $dp = d\ mod\ p-1$ and $dq = d\ mod\ q-1$. It is then possible to recover the plaintext M using the Chinese remainder theorem. This method is faster because it computes two exponentiations of n/2- bit integers instead of one exponentiation of n-bit integers [1].

Algorithm: RSA decryption using CRTInput: C, p, q, N;
Output: M;

1. dp ←d mod( p - 1);
2. dq ←d mod(q - 1);
3. Mp ← $C^{dp}$ mod p;
4. Mq← $C^{dq}$ mod q;
5. pinv ← $p^{-1}$ mod q;
6. M ←CRT(Mp ,Mq, p, q, (pinv), N);
7. return(M); [1]

RSA-CRT decryption Algorithm shows an optimization of RSA using the Chinese remainder theorem and the fast exponentiation algorithm. The two inverses that are normally needed for the Chinese remainder theorem are reduced in one inversion, which is pre-computed.

## 3. Complexity of RSA-CRT algorithm

The private key in RSA-CRT is <dp, dq>, the key length of dp and dq are (n/2). So RSA-CRT require twotimes $O((n)/2)^3)$ compared to the $O(n^3)$ decryption in RSA standard.
Thus, the theoretical speedup of RSA-CRT is

$$\text{RSA-CRT} = n^3/2(n/2)^3 = 4$$

Then, the decryption procedure of RSA-CRT is about 4 times faster than that of RSA standard

**Rebalanced RSA-CRT**

Based on the Chinese Remainder Theorem (CRT), Wiener suggested another RSA variant, Rebalanced RSA-CRT, to make the decryption time is much faster by carefully choosing *d* in the key generation phase such that $dp \equiv d \pmod{p-1}$ and $dq \equiv d \pmod{q-1}$ are small.
One first selects two small CRT-exponents *dp* and *dq*, and then these two CRT-exponents are combined to get the secret exponent. At last, computes the corresponding public exponent ***e*** satisfying $ed \equiv 1 \pmod{\varphi(N)}$ [5]. This variant of RSA enables rebalances the costs of encryption and decryption. In other words, it speeds up the RSA decryption by shifting the decryption cost to the encryption cost.

In Rebalanced RSA-CRT, both d and e will be of the same order of magnitude as φ (N). The decryption time depends on the bit-size of dp and dq, while not on the bit-size of d, the decryption time is minimized. But the encryption time depends on the bit-size of **e**, this will make the encryption for the original Rebalanced RSA-CRT very time-consuming.
The algorithm takes two security parameters as input: n (typically 1024) and k (typically 160) where k < n/2. It next picks two primes p and q verifying gcd (p−1, q−1) = 2 and whose bit length is n/2. The modulus N is N = pq. It then randomly picks two k-bit values dp and dq such that gcd (dp, p−1) = 1, gcd (dq, q−1) = 1 and dp = dq mod 2. The secret exponent d must verify: $d = dp \pmod{p-1}$ and $d = dq \pmod{q-1}$. Cannot directly compute d with the Chinese remainder theorem because p−1 and q−1 are not relatively prime (they are both even). But we chose them such that gcd (p −1, q−1) = 2, therefore:

$$gcd\left(\frac{p-1}{2}, \frac{q-1}{2}\right) = 1$$

We also know that $dp = dq \pmod 2$. To compute the public exponent e, we just have to compute the inverse of *d* modulo Ø (N) = (p − 1) (q − 1). This is allowed because gcd (dp, p − 1) = gcd (*dq,q−1*) = 1. Therefore, gcd (d, p −1) = gcd (d, q−1) = 1. And finally gcd (d, (p −1)(q−1)) = 1. We have no control over ***e*** which is of the order of *N*. The encryption won't be as fast as in standard RSA but we manage to increase the speed of the decryption stage.
Now we describe three constituent algorithms: key generation, encryption, and decryption.

## 1. Rebalanced RSA-CRT Key generation:

The key generation of the Rebalanced RSA is as follows:

Algorithm: Key Generation in
Rebalanced RSA-CRTInput: N, k;
Output: p, q, dp, dq, e, d;

1. Randomly select two 512-bit primes p= 2p1+1 and q= 2q1+1 such that gcd (p1, q1) = 1.
2. Compute $p1inv = p1^{-1} \pmod{q1}$
3. Randomly select two distinct odd numbers dp and dq of 160 bits such that gcd (dp, p−1) =1 and

gcd (dq,q − 1) = 1.

4. Compute d = CRT (dp, dq, p1, q1,$_{p1\ inv,\ N}$).

5. Compute e ≡ $d^{-1}$ mod (p− 1) (q − 1).

6. Return (p, q, dp, dq, e, d). [5]

The public key is given by the pair *(e, N)* and the private key is given by the tuple *(dp, dq, p, q)*. Note that *e* will be roughly the same order of magnitude as *Ø(N)*. Thus, in order to reduce the decryption time even further than RSA-CRT, the encryption time is essentially maximized.

**2. Rebalanced RSA-CRT Encryption**

This is exactly the same as in standard RSA, that is $C \equiv (mod\ N)$, except that e is much larger. The public key is (N, e). The complexity of this algorithm Take time $O(e\ M(n))$,where $M(n)$ is the cost of multiplying two n-bit integers and can be taken as $O(n^2)$.And $|e| \approx |n|$, thus these algorithms take time $O(n^3)$,compareto $O(n^2)$in RSA Standard

**3. Rebalanced RSA-CRT Decryption**

The decryption is the same as the CRT decryption. The main difference is that in Rebalanced RSA-CRT, the CRT-exponents *dp* and *dq* are only of *160* bits which are much shorter than the CRT exponents of *512* bits in RSA-CRT. Thus, the decryption in Rebalanced RSA-CRT is about *512/160 = 3.2* times faster than that in RSA-CRT.

The private key is *(p, q, dp, dq)*. Can decrypt a cipher text *C* by Using the Chinese remainder theorem, we are then able to recover the plaintext M, verifying M = Mp mod P and M = Mq mod q. as follows: Algorithm: Rebalanced RSA-CRT Decryption

Input: N, k;

Output: C, N, dp, dq, p, q;

1. Mp ← $C^{dp}$ mod p;
2. Mq ← $C^{dq}$ mod q;
3. pinv ← $P^{-1}$ mod p;
4. M ←CRT(Mp, Mq, p, q, (pinv), N);
5. Return(M); [9]

**Complexity of Rebalanced RSA-CRT algorithm**

Recall that dp, dq in Rebalanced RSA are k-bits each. The cost of modular multiplications is the same as that of RSA-CRT; the main difference is the number of multiplications computed during each modular exponentiation.

The decryption in Rebalanced RSA require tow times $O(k(n/2)^2)$. Compared to the $O(n^3)$ decryption ofthe RSA standard, then a theoretical speedup of Rebalanced RSA is

Rebalanced RSA-CRT = $n^3/2k(n/2)^2$ = 2n/k

So, for modulo of 1024 bits with k = 160, Rebalanced RSA is theoretically 12.8 Faster than RSA standardand 3.2 faster than RSA-CRT.

## IMPROVING REBALANCED RSA-CRT ENCRYPTION TIME

According to the key generation in the original Rebalanced RSA-CRT, if we first select the small CRT- exponents *dp* and *dq*, the public exponent *e* will be of the same bit-size as modulus *Ø(N)*. This causes heavy encryption cost. If we can make the public exponent *e* much shorter than *Ø(N)*, it will be more convenient and practical in many applications.

Two variants of Rebalanced RSA-CRT, Scheme A and Scheme B have been design to achieve the above goal [5]. The two key generation algorithms for Rebalanced RSA-CRT which generates public

exponents much smaller than *Ø(N)*. Each key generation algorithm is based on the following fundamental theorem from number theory.

**Theorem 3.1**: If a and b are relatively prime, i.e. *gcd(a, b) = 1*, then we can find a unique pair *(u , v )*

satisfying $au - bv = 1$, for any integer $u,v \in z$ [7] .

**A. Rebalanced RSA-CRT Scheme A**

The first key generation algorithm, scheme A, produces a *568-bit* public exponent, e.g., $e = 2^{567} + 1$, andtwo 160-bit CRT-exponents *dp, dq*. The key generation of the algorithm is as follows:

Algorithm: Key Generation in Scheme AInput: *n, k;*

Output *dp, dq, p, q, e;*

1. Randomly select an odd number ***e*** of *568* bits.
2. Randomly select a number *kp1* of *160* bits, such that *gcd (kp1, e) = 1*.
3. Based on Theorem 3.1, we can uniquely determine two numbers *dp, kp1 < dp < 2 kp1*, and P, *e < P <2e*, satisfying *e dp = kp1 P + 1*.
4. Factor *P* as $P = kp2 \cdot p$̀ such that *kp2* is a number of *56* bits and p= (p̀ +1) is a prime number. If this isinfeasible, then go to Step 2.
5. Randomly select a number *kq1* of *160* bits, such that gcd (*kq1*, e) = 1.
6. Based on Theorem 3.1, we can uniquely determine two numbers *dq,kq1 < dq <2 kq1*, and q, *e < q < 2e*,satisfying *e dq = kq1 q + 1*.
7. Factor *q* as $q = kq2 \cdot q$̀ such that *kq2* is a number of *56* bits and q= (q´ +1) is a prime number. If thisis infeasible, then go to Step 5.
8. Return *(dp, dq, p, q, e)*. [5]

## Complexity of Rebalanced RSA-CRT Scheme A

The complexity of encryption in Rebalanced is the same as the complexity of decryption in RSA, because the public exponent *e* will be of the same bit-size as modulus *Ø(N)*.then *C* = mod *N* take $1.5\ n^3 = O(n^3)$. Scheme A, use a *512*-bit public exponent, thus Scheme A take $k.n^2 = O(kn^2)$, where k is bit length of public exponent. The theoretical speedup of Scheme A is

Schema A = $1.5\ n^3/\ kn^2 = 1.5\ n/k$

So, for modulo of 1024 bits with k = 568, Scheme A is theoretically 2.7 Faster than Rebalanced RSA-CRT.

## B. Rebalanced RSA-CRT Scheme B

The second key generation algorithm, called Scheme B. In Scheme B, The Key Generation produces a *512*- bit public exponent, e.g., $e = 2^{511} + 1$, and two *198-bit* CRT-exponents *dp, dq*. The encryption is about *3* times faster than that of Rebalanced RSA-CRT [5].

Algorithm: Key Generation in Scheme BInput: *n, k;*

Output *dp, dq, p, q, e;*

1. Randomly select an odd number *e* of *512* bits.
2. Randomly select an odd number *kp* of *198* bits, such that *gcd (kp, e) = 1*.
3. Based on Theorem 1, we can uniquely determine two numbers dp, kp < dp < 2kp, and p´, e < p´ < 2e, satisfying *e.dp – kp.p´ = 1*.

4. If $p = p' + 1$ is not a prime number, then go to Step 2.
5. Randomly select an odd number $kq$ of *198* bits, such that $gcd(kq, e) = 1$.
6. Based on Theorem 3.1, we can uniquely determine two numbers dq, $kq < dq < 2kq$, and $q'$, $e < q' < 2e$, satisfying $e\,dq - kq\,q' = 1$.
7. If $q = q' + 1$ is not a prime number, then go to Step 5.
8. The public key is *(N, e);* the secret key is *(dp, dq, p, q).* [5]

The key generation algorithm for this scheme is much more efficient than the one in Scheme A. The runtime of the algorithm is dominated by two loops, like Scheme A, but each iteration requires much less computational effort as there is no factoring required [7].

## 1. Complexity of RSA-CRT Rebalanced Scheme B

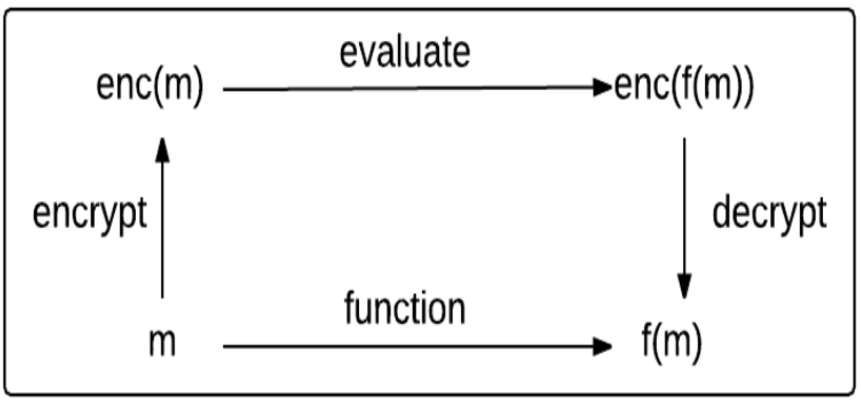
The Complexity of Scheme B the same as in Scheme A, the main different is bit length of *e* = *512*

$$Scheme\ B = \frac{1.5n^3}{kn^2} = 1.5\ n/k$$

So, for modulo of 1024 bits with k = 512, Scheme B is theoretically 3 times faster than Rebalanced RSA- CRT. But the decryptions in two Schemes are a little slower than that of Rebalanced RSA-CRT.

## ➢ Homomorphic Encryption techniques

**Homomorphic encryption** is a cryptographic method that allows mathematical operations on data to be carried out on cipher text, instead of on the actual data itself. The cipher text is an encrypted version of the input data (also called plain text). It is operated on and then decrypted to obtain the desired output. The critical property of homomorphic encryption is that the same output should be obtained from decrypting the operated cipher text as from simply operating on the initial plain text.




The following graphic explains how homomorphism is used in encryption. The process begins with some plain text message, mm. It could be the name of someone in a computer system, "jane doe". The goal is to perform some function ff on it, like capitalizing the name. That's possible, but it would be safer to encrypt the message using enhance before performing any functions on it so the person performing the function never learns Jane's name. So the message is encrypted to some cipher text 163726. Then, it is evaluated, or transformed, into another value using some other

function, f(x) = x + 127f(x)=x+127, for example. The output, 163853, is another completely encrypted message. This message can then be decrypted to "Jane Doe".

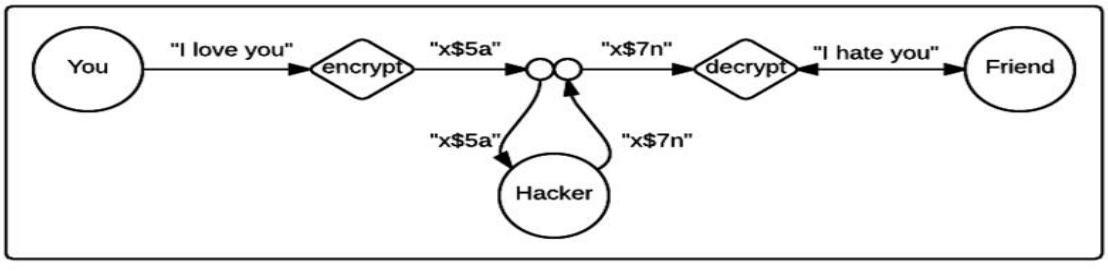
Current homomorphic encryption exists in partial and full forms. Partially homomorphic systems only allow certain operations on encrypted data, typically multiplication or division, and have existed for many years. Fully homomorphic systems, which allow all operations on encrypted data, remained an open problem in cryptography for 30 years, but they were finally solved in 2009. It is typically desirable because it allows a system to chain together multiple operations that are all performed on encrypted data so as not to expose unencrypted data.

In cryptography, it is assumed that there is some sort of communication going on between two people. Let's call them Alice and Bob. Maybe Alice wants to give Bob a suitcase full of money to count and give back to her. Alice hands Bob the suitcase, and Bob asks for the key to open it. Alice doesn't want to give him the key because she's not very trusting. Bob, however, doesn't know how to count the money if Alice won't let him open the box. So, the couple have two options:

1. Alice could give Bob the key. This would work. However, there is also an attacker out there who wants to steal the money, Eve. She could be impersonating Bob, or she could steal both the suitcase and key from Bob later.
2. Devise a new kind of lock, even better than a suitcase that needs a key. This lock won't let anyone who isn't Alice get to the money or even know how much money is in there. But it will allow anyone with the right permission to count it. This is the concept behind homomorphic encryption and why it's so powerful.

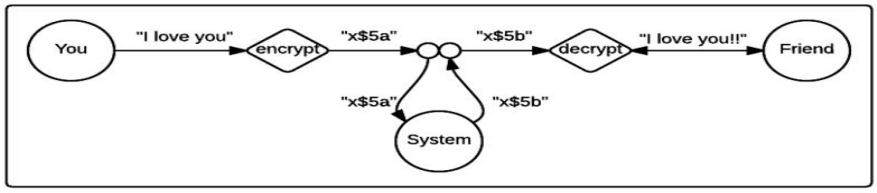
Properties

Homomorphic encryption is **malleable** by design. A malleable crypto-system is one in which anyone can intercept a cipher text, transform it into another cipher text, and then decrypt that into a plain text that makes sense. Malleability is generally considered undesirable in a crypto-system. Imagine you're trying to send the message "I love you" to your friend using encryption. You encrypt it and send it off. But, it is intercepted by a hacker on the way. All they see is some cipher text, but they can change that cipher text to something that will decrypt to "I hate you" when your friend tries to decrypt it. That is why malleability to not usually wanted.




A malleable crypto-system

However, homomorphic systems should be malleable. Maybe You want to build a system that simply adds exclamation marks to whatever you send your friend. But, you don't want the system to know what you're sending your friend, that's a secret. You encrypt your message and send it along to your system. Your system only sees the cipher text, so it doesn't know what you're saying, but it can still transform the cipher text. It transforms the cipher in a text that will decrypt to the same input message, plus a few exclamation marks. So, now it looks like this.




Another malleable crypto-system

Malleable systems allow for multiple parties, especially in cloud-based environments, to operate on data without ever exposing it.

## Applications

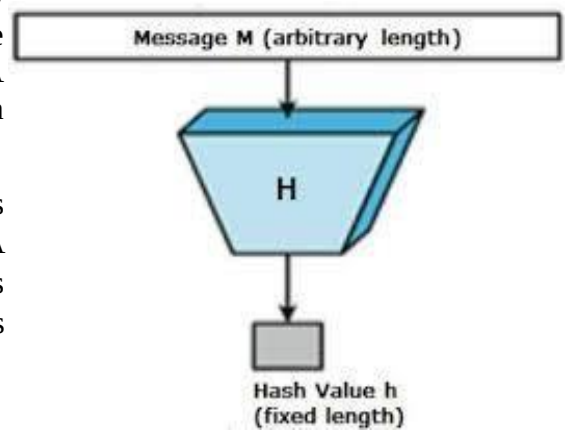
Homomorphic encryption has seen revitalization in recent years, especially due to cloud computing. Resources for computation are so cheap that many system outsource their computation to companies like Amazon or Microsoft and their huge data centers. However, those systems don't want to expose that data for attack. So, they encrypt the data, send it to a data center, it is operated on, and then it is send back to the original system for decryption.

Privacy is very important in today's computing environment. Private information that is used by various systems often uses homomorphic encryption if that data needs to stay encrypted. E-voting, for example, might want to use homomorphic encryption to protect voting privacy.

E-cash systems might also use homomorphic encryption. If you send money to a friend, you might not want the system to know how much money you are sending. The system could retrieve both you and your friend's encrypted data, add them together, and then send them back to your friend.

## ➢ Message Authentication and Hash Functions

A **hash function** H accepts a variable-length block of data M as input and produces a fixed-size hash value h = H(M). A "good" hash function has the property that the results of applying the function to a large set of inputs will produce outputs that are evenly distributed and apparently random. In general terms, the principal object of a hash function is data integrity. A change to any bit or bits in M results, with high probability, in a change to the hash value.




The kind of hash function needed for security applications is referred to as a **cryptographic hash function**. A cryptographic hash function is an algorithm for which it is computationally infeasible (because no attack is significantly more efficient than brute force) to find either

(a) a data object that maps to a pre-specified hash result (the one-way property) or

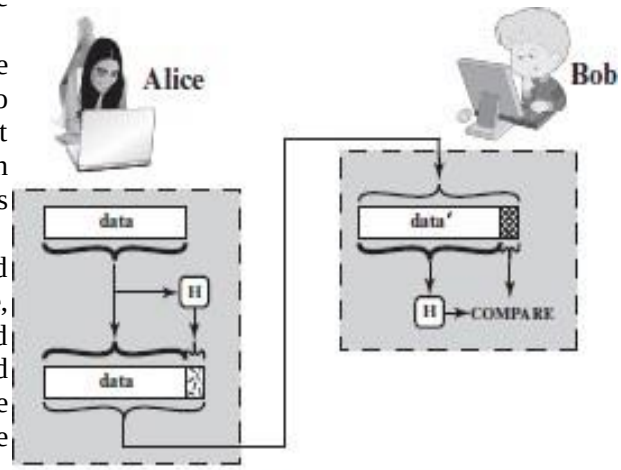
(b) two data objects that map to the same hash result (the collision-free property).

Because of these characteristics, hash functions are often used to determine whether or not data has changed.

Figure: Cryptographic Hash Function; h = H(M)

## Message Authentication:

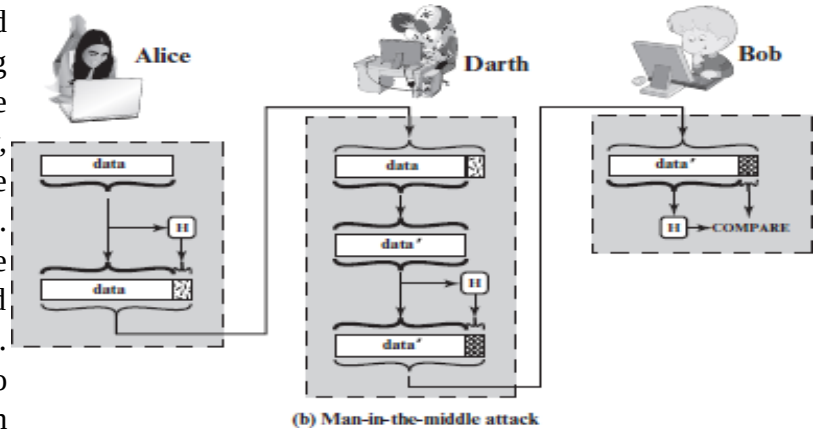
- Message authentication is a mechanism or service used to verify the integrity of a message.
- Message authentication assures that data received are exactly as sent (i.e., there is no modification, insertion, deletion, or replay).
- In many cases, there is a requirement that the authentication mechanism assures that purported identity of the sender is valid.
- When a hash function is used to provide message authentication, the hash function value is often referred to as a message digest.
- The essence of the use of a hash function for message integrity is as follows.
- The sender computes a hash value as a function of the bits in the message and transmits both the hash value and the message. The receiver performs the same hash calculation on the message bits and compares this value with the incoming hash value.
- If there is a mismatch, the receiver knows that the message (or possibly the hash value) has beenaltered.
- The hash value must be transmitted in a secure fashion. That is, the hash value must be protected so that if an adversary alters or replaces the message, it is not feasible for adversary to also alter the hash value to fool the receiver. This type of attack is shown in Figure (b).
- In this example, Alice transmits a data block and attaches a hash value. Darth intercepts the message, alters or replaces the data block, and calculates and attaches a new hash value. Bob receives the altered data with the new hash value and does not detect the change. To prevent this attack, the hash value generated by Alice must be protected.



(a) Use of hash function to check data integrity

Figure below illustrates a variety of ways in which a hash code can be used to provide message authentication, as follows.

a) The message plus concatenated hash code is encrypted using symmetric encryption. Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.



(b) Man-in-the-middle attack

b) Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

c) It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value S. A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B

possesses S, it can recompute the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.

d) Confidentiality can be added to the approach of method (c) by encrypting the entire message plus the hash code.

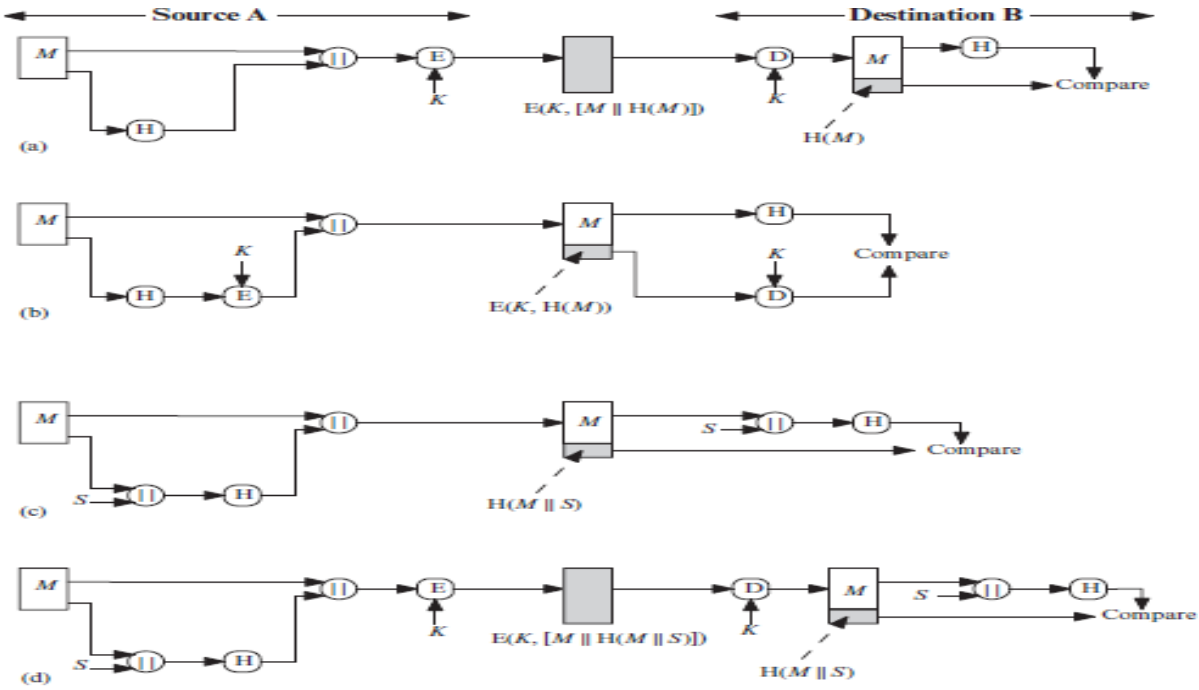



Figure 11.3 Simplified Examples of the Use of a Hash Function for Message Authentication

When confidentiality is not required, method (b) has an advantage over methods (a) and (d), which encrypts the entire message, in that less computation is required. Nevertheless, there has been growing interest in techniques that avoid encryption.

Several reasons for this interest are pointed out as:

- Encryption software is relatively slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.
- Encryption hardware costs are not negligible. Low-cost chip implementations of DES are available,but the cost adds up if all nodes in a network must have this capability.
- Encryption hardware is optimized toward large data sizes. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead.
- Encryption algorithms may be covered by patents, and there is a cost associated with licensingtheir use.

## Message Authentication Functions:

The message authentication function is concerned with the types of functions that may be used to pro- duce an authenticator. These may be grouped into three classes:

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

## Message Authentication Code (MAC):

A authentication technique that involves the use of a secret key to generate a small fixed-size block of data that is appended to the message is known as Message Authentication Code (MAC). This technique assumes that two communicating parties, say A and B, share a common secret key K. When A has a message to send to B, it calculates the MAC as a function of the message and the key:
MAC = C (K, M)
where
M = input messageC = MAC function
K = shared secret key
MAC = message authentication code
The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC.

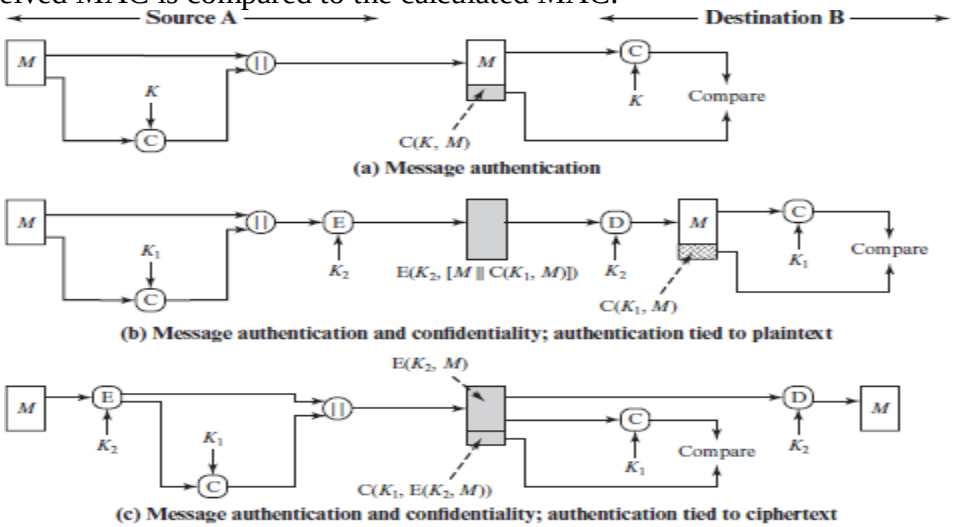



Figure 12.4 Basic Uses of Message Authentication code (MAC)

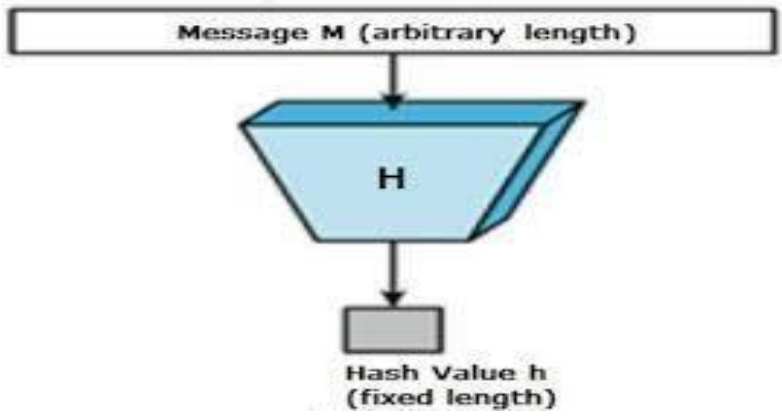
If we assume that only the receiver and the sender know the identity of the secret key, and if the receivedMAC matches the calculated MAC, then:

- The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
- The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.
- If the message includes a sequence number (such as is used with TCP), then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

A MAC function is similar to encryption. One difference is that the MAC algorithm need not be reversible, as it must be for decryption. In general, the MAC function is a many-to-one function. The domain of the function consists of messages of some arbitrary length, whereas the range consists of all possible MACs and all possible keys.

## Cryptographic Hash Function:

- A cryptographic hash function (CHF) is a hash function that is suitable for use in cryptography.
- It is a mathematical algorithm that maps data of arbitrary size (often called the "message") to a bit string of a fixed size (the "hash value", "hash", or "message digest") and is a one-way function, that is, a function which is practically infeasible to invert.



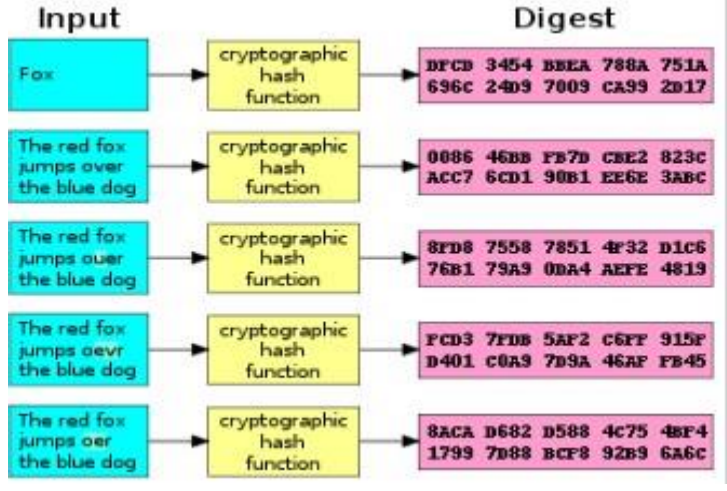

- Ideally, the only way to find a message that produces a given hash is to attempt a brute-force search of possible inputs to see if they produce a match, or use a table of matched hashes.
- Cryptographic hash functions are a basic tool of modern cryptography.

Figure: Cryptographic Hash Function; h = H(M)

## Properties of Hash Function:

The ideal cryptographic hash function has the following main properties:

- it is deterministic, meaning that the same message always results in the same hash
- it is quick to compute the hash value for any given message
- it is infeasible to generate a message that yields a given hash value (Pre-Image Resistant)
- it is infeasible to find two different messages with the same hash value (Collision Resistance)
- A small change to a message should change the hash value so extensively that the new hash valueappears uncorrelated with the old hash value (avalanche effect)




## Applications of Hash Function:

Given that the hash depends on the input to the hash function and will change with the input hash functions are used:

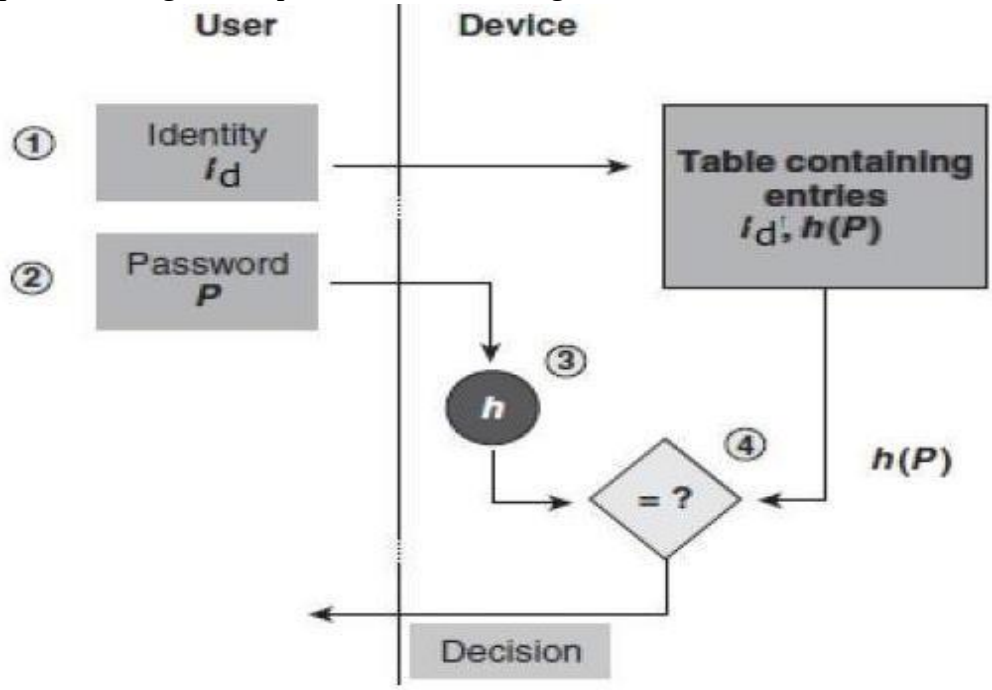
- to ensure that messages have not been tampered with (message authentication, digital signatures, checksums) and
- to check for equality while preserving secrecy / efficiently (for example, checking if password is correct without storing the actual password, or checking for duplicates in lists of large items).
- They are also used as proof-of-work (for example, in cryptocurrencies like bitcoin), error- correcting codes, and randomization and to make cryptographic algorithms more efficient.

There are two direct applications of hash function based on its cryptographic properties.

Password Storage

Hash functions provide protection to password storage.

- Instead of storing password in clear, mostly all logon processes store the hash values of passwords in the file.
- The Password file consists of a table of pairs which are in the form (user id, h(P)).
- The process of logon is depicted in the following illustration:

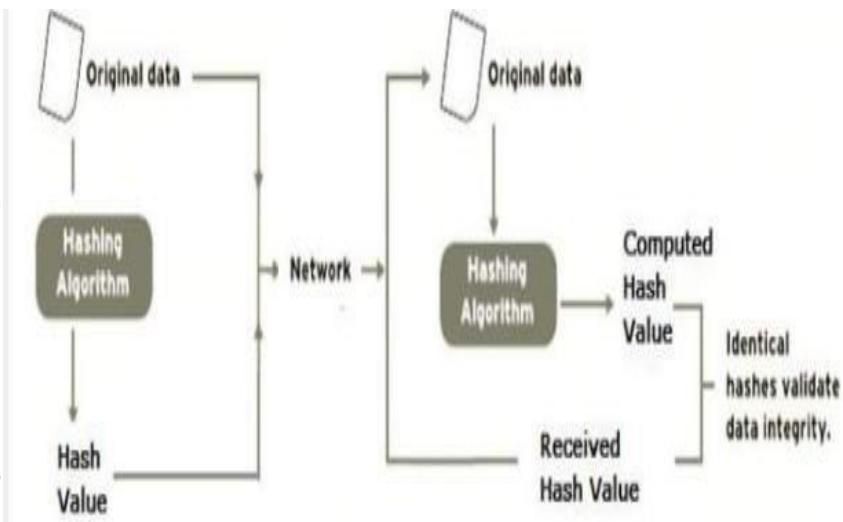



- An intruder can only see the hashes of passwords, even if he accessed the password. He can neither logon using hash nor can he derive the password from hash value since hash function possesses the property of pre-image resistance.

Data Integrity Check

Data integrity check is a most common application of the hash functions. It is used to generate the checksums on data files. This application provides assurance to the user about correctness ofthe data.

The process is depicted in the following illustration:




The integrity check helps the user to detect any changes made to original file. It however, does not provide any assurance about originality. The attacker, instead of modifying file data, can change the entire file and compute all together new hash and send to the receiver. This integrity check application is useful only if the user is sure about the originality of file.

- **Hash algorithms**

A cryptographic hash function is an algorithm that takes an arbitrary amount of data input—a credential—and produces a fixed-size output of enciphered text called a hash value, or just "hash." That enciphered text can then be stored instead of the password itself, and later used to verify the user.
Certain properties of cryptographic hash functions impact the security of password storage.

- **Non-reversibility, or one-way function.** A good hash should make it very hard to reconstruct the original password from the output or hash.
- **Diffusion, or avalanche effect**. A change in just one bit of the original password should result in change to half the bits of its hash. In other words, when a password is changed slightly, the output of enciphered text should change significantly and unpredictably.
- **Determinism.** A given password must always generate the same hash value or enciphered text.
- **Collision resistance.** It should be hard to find two different passwords that hash to the same enciphered text.
- **Non-predictable.** The hash value should not be predictable from the password.

There are variations that can improve your hash function and provide a greater barrier against attacks.

**Salted hashes**

Salting adds random data to each plaintext credential. The result: two identical plaintext passwords are now differentiated in enciphered text form so that duplicates cannot be detected.

**Keyed hash functions**

A keyed hash function (also known as a hash message authentication code, or HMAC) is an algorithm that uses a cryptographic key AND a cryptographic hash function to produce a message authentication code that is keyed and hashed.

**Adaptive hash functions**

An adaptive one-way function is any function that is designed to iterate on its inner workings, feeding the output back as input, in a manner that causes it to–ultimately–take longer to execute. It is adaptive because the developer can adjust how many iterations occur. To protect stored passwords, architects have applied the adaptive design to hash functions (such as PBKDF2) and to encryption schemes.

**The trade-off of cryptographic hash functions**

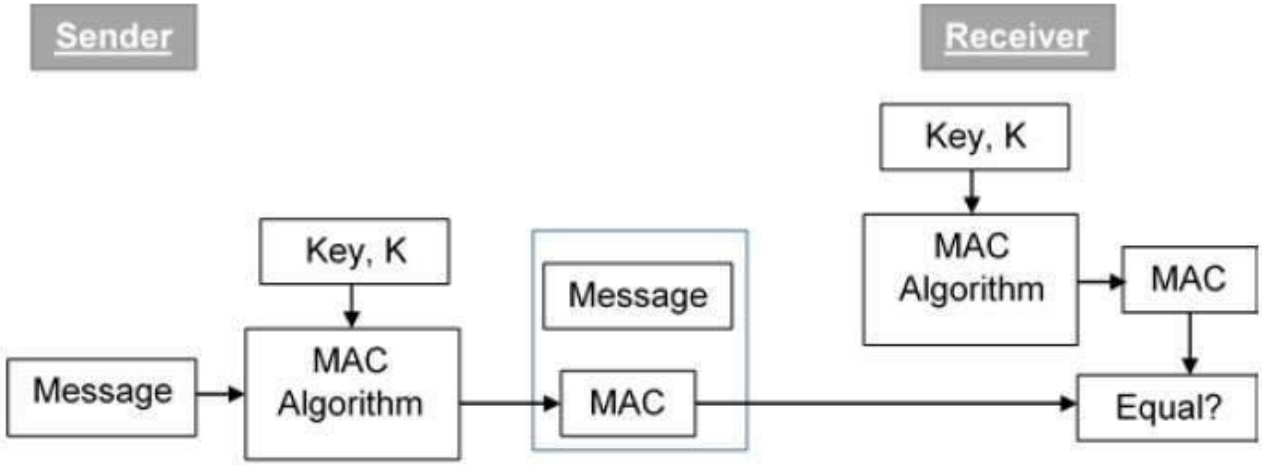
Cryptographic hash functions do provide barriers to attackers, like speed bumps slowing down a speeding motorcycle. But it's critical to remember that eventually the motorcycle will still make it down the street. However, these barriers will slow down your defenders as well—normal users and your server. Set the speed bump too high, and you run the risk of annoying your user—and overtaxing your server.

## ➢ Message Authentication Code (MAC)

MAC algorithm is a symmetric key cryptographic technique to provide message authentication. For establishing MAC process, the sender and receiver share a symmetric key K.
Essentially, a MAC is an encrypted checksum generated on the underlying message that is sent along with a message to ensure message authentication.
The process of using MAC for authentication is depicted in the following illustration –
Let us now try to understand the entire process in detail −

- The sender uses some publicly known MAC algorithm, inputs the message and the secret key K and produces a MAC value.
- Similar to hash, MAC function also compresses an arbitrary long input into a fixed length output. The major difference between hash and MAC is that MAC uses secret key during the compression.

- The sender forwards the message along with the MAC. Here, we assume that the message is




sent in the clear, as we are concerned of providing message origin authentication, not confidentiality. If confidentiality is required then the message needs encryption.

- On receipt of the message and the MAC, the receiver feeds the received message and the shared secret key K into the MAC algorithm and re-computes the MAC value.
- The receiver now checks equality of freshly computed MAC with the MAC received from the sender. If they match, then the receiver accepts the message and assures himself that the message has been sent by the intended sender.
- If the computed MAC does not match the MAC sent by the sender, the receiver cannot determine whether it is the message that has been altered or it is the origin that has been falsified. As a bottom-line, a receiver safely assumes that the message is not the genuine.

Limitations of MAC

There are two major limitations of MAC, both due to its symmetric nature of operation −

- **Establishment of Shared Secret.**
  - It can provide message authentication among pre-decided legitimate users who have shared key.
  - This requires establishment of shared secret prior to use of MAC.
- **Inability to Provide Non-Repudiation**
  - Non-repudiation is the assurance that a message originator cannot deny any previously sent messages and commitments or actions.
  - MAC technique does not provide a non-repudiation service. If the sender and receiver get involved in a dispute over message origination, MACs cannot provide a proof that a message was indeed sent by the sender.
  - Though no third party can compute the MAC, still sender could deny having sent the message and claim that the receiver forged it, as it is impossible to determine which of the two parties computed the MAC.

**MACs based on hash functions: HMAC**

**HMAC Design Objectives**

RFC 2104 lists the following design objectives for HMAC.

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

secure hashfunctions are found or required.

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

## HMAC Algorithm

H = embedded hash function (e.g., MD5, SHA-1, RIPEMD-160)

*IV* = initial value input to hash function

*M* = message input to HMAC (including the padding specified in the embeddedhash function)

*Yi* _ *i* th block of *M*, 0 ≤*i* ≤(*L* – 1)

*L* _ number of blocks in *M*

*b* _ number of bits in a block

*n* _ length of hash code produced by embedded hash function

*K* _ secret key; recommended length is ≥*n*; if key length is greater than *b*, the key is input to the hash functionto produce an *n*-bit key

*K*+ _ *K* padded with zeros on the left so that the result is *b* bits in lengthipad _ 00110110 (36 in hexadecimal) repeated *b*/8 times

opad _ 01011100 (5C in hexadecimal) repeated *b*/8 times

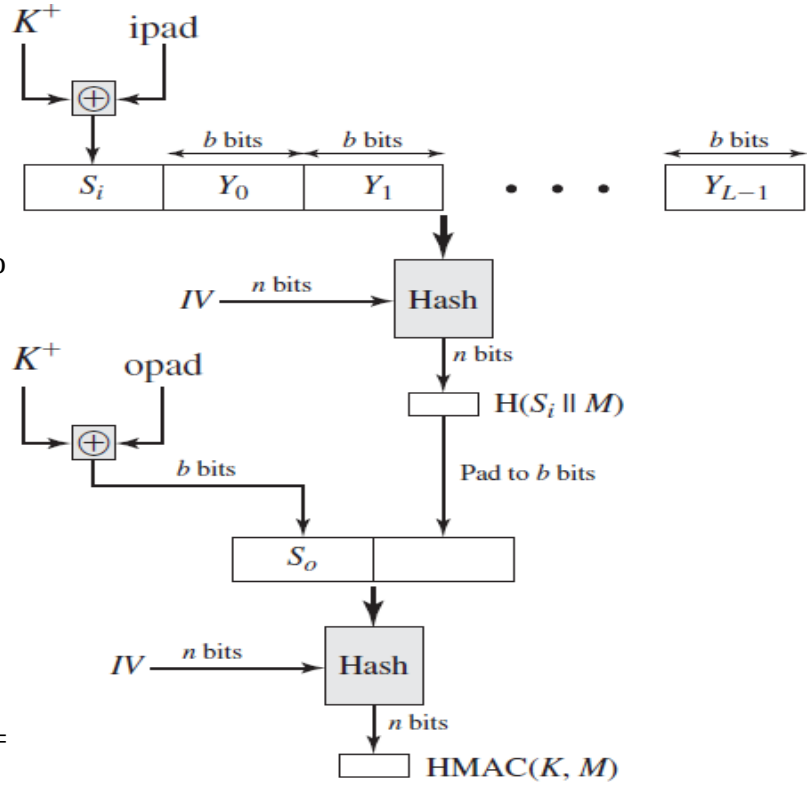



fig: HMAC Structure

HMAC can be expressed as: HMAC(*K*, *M*) = H[(*K*+ ⊕ opad) || H[(*K*+ ⊕ ipad) || *M*]]

The algorithm is as follows:

**1.** Append zeros to the left end of K to create a b-bit string $K^+$ (e.g., if K is of length 160 bits and , then will be appended with 44 zeroes).

**2.** XOR (bitwise exclusive-OR) with ipad to produce the b-bit block $S_i$.

**3.** Append M to $S_i$.

**4.** Apply H to the stream generated in step 3.

**5.** XOR $K^+$ with opad to produce the b-bit block $S_O$

**6.** Append the hash result from step 4 to $S_O$.

**7.** Apply H to the stream generated in step 6 and output the result.

### ➢ RSA Encryption Algorithm

RSA encryption algorithm is a type of public-key encryption algorithm. To better understand RSA, lets first understand what is public-key encryption algorithm.

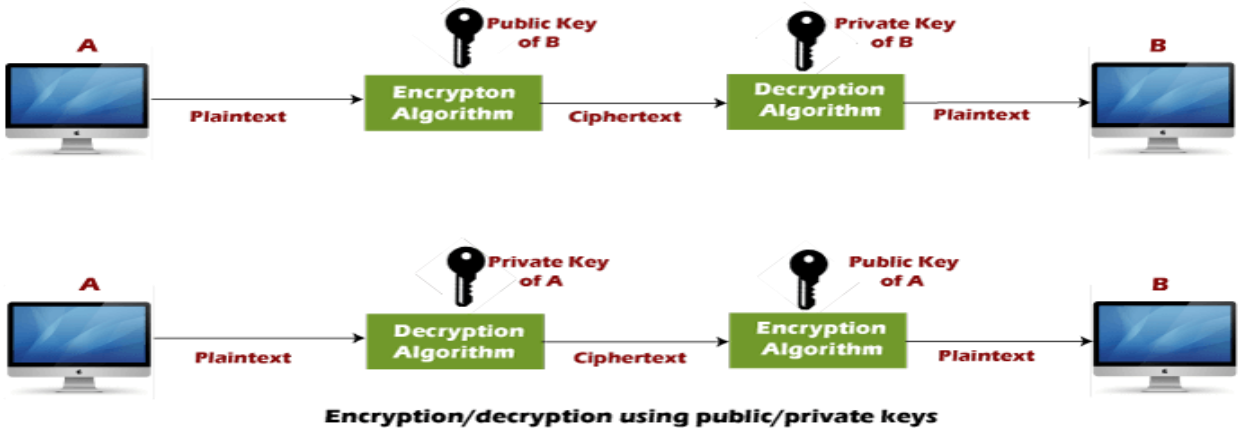
Public key encryption algorithm:

Public Key encryption algorithm is also called the Asymmetric algorithm. Asymmetric algorithms are those algorithms in which sender and receiver use different keys for encryption and decryption. Each sender is assigned a pair of keys:

- **Public key**
- **Private key**

The **Public key** is used for encryption, and the **Private Key** is used for decryption. Decryption cannot be done using a public key. The two keys are linked, but the private key cannot be derived from the public key. The public key is well known, but the private key is secret and it is known only to the user who owns the key. It means that everybody can send a message to the user using user's public key. But only the user can decrypt the message using his private key.

The Public key algorithm operates in the following manner:

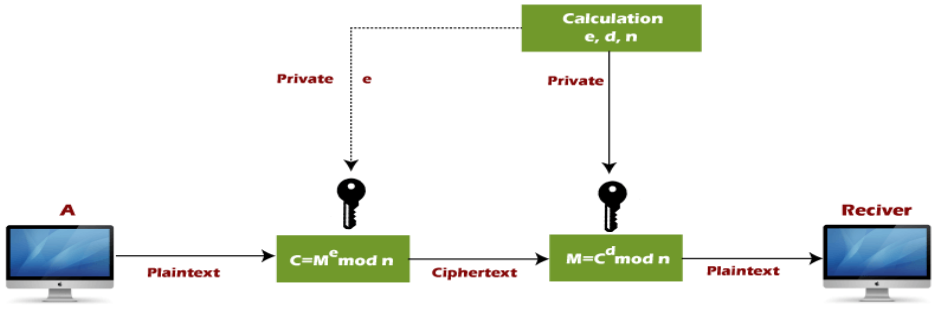



**Encryption/decryption using public/private keys**

- The data to be sent is encrypted by sender **A** using the public key of the intended receiver
- B decrypts the received ciphertext using its private key, which is known only to B. B replies to A encrypting its message using A's public key.
- A decrypts the received ciphertext using its private key, which is known only to him.

RSA encryption algorithm:

RSA is the most common public-key algorithm, named after its inventors **Rivest, Shamir, and Adelman (RSA).**




**RSA**

**RSA algorithm uses the following procedure to generate public and private keys:**

- Select two large prime numbers, p and **q**.

- Multiply these numbers to find **n = p x q,** where **n** is called the modulus for encryption and decryption.
- Choose a number **e** less than **n**, such that n is relatively prime to **(p - 1) x (q -1).** It means that **e** and **(p - 1) x (q - 1)** have no common factor except 1. Choose "e" such that 1<e < φ (n), e is prime to φ (n), **gcd (e,d(n)) =1**
- If **n = p x q,** then the public key is <e, n>. A plaintext message **m** is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C. **$C = m^e \bmod n$**
- Here**, m** must be less than **n**. A larger message (>n) is treated as a concatenation of messages, each of which is encrypted separately.
- To determine the private key, we use the following formula to calculate the d such that: **$D_e \bmod \{(p - 1) \times (q - 1)\} = 1$**
- **Or**
  **$D_e \bmod \varphi(n) = 1$**
- The private key is <d, n>. A ciphertext message **c** is decrypted using private key <d, n>. To calculate plain text **m** from the ciphertext c following formula is used to get plain text m. **$m = c^d \bmod n$**

Let's take some example of RSA encryption algorithm:

Example 1:

This example shows how we can encrypt plaintext 9 using the RSA public-key encryption algorithm. This example uses prime numbers 7 and 11 to generate the public and private keys.

**Explanation:**

**Step 1:** Select two large prime numbers, p, and **q**.

p = 7

q = 11

**Step 2:** Multiply these numbers to find **n = p x q,** where **n** is called the modulus for encryption and decryption.

First, we calculate

**n = p x q**

n = 7 x 11

n = 77

**Step 3:** Choose a number **e** less that **n**, such that n is relatively prime to **(p - 1) x (q -1).** It means that **e** and **(p - 1) x (q - 1)** have no common factor except 1. Choose "e" such that 1<e < φ (n), e is prime to φ (n), gcd (e, d (n)) =1.

Second, we calculate

**φ (n) = (p - 1) x (q-1)**

φ (n) = (7 - 1) x (11 - 1)

φ (n) = 6 x 10

φ (n) = 60

Let us now choose relative prime e of 60 as 7.

Thus the public key is <e, n> = (7, 77)

**Step 4:** A plaintext message **m** is encrypted using public key <e, n>. To find ciphertext from the plain text following formula is used to get ciphertext C.

To find ciphertext from the plain text following formula is used to get ciphertext C.

**$C = m^e \bmod n$**

$C = 9^7 \bmod 77$

C = 37

**Step 5:** The private key is <d, n>. To determine the private key, we use the following formula d such that:

**$D_e$ mod {(p - 1) x (q - 1)} = 1**

7d mod 60 = 1, which gives d = 43

The private key is <d, n> = (43, 77)

**Step 6:** A ciphertext message **c** is decrypted using private key <d, n>. To calculate plain text **m** from the ciphertext c following formula is used to get plain text m.

**$m = c^d$ mod n**

$m = 37^{43}$ mod 77

m = 9

In this example, Plain text = 9 and the ciphertext = 37

UNIT-IV

## ➢ Digital Signatures and authentication Protocols:

The digital signature must have the following properties:

■ It must verify the author and the date and time of the signature.

■ It must authenticate the contents at the time of the signature.

■ It must be verifiable by third parties, to resolve disputes. Thus, the digital signature function includes the authentication function.

**Digital Signature Requirements**

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

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

■ The signature must use some information only known to the sender to prevent both forgery and denial.

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

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

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

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

**Direct Digital Signature**

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

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

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

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

**ELGAMAL DIGITAL SIGNATURE SCHEME**

Before examining the NIST Digital Signature Algorithm, it will be helpful to under- stand the Elgamal and Schnorr signature schemes. Recall from Chapter 10, that the Elgamal encryption scheme is designed to enable encryption by a user's public key with decryption by the user's private key. The Elgamal signature scheme involves the use of the private key for digital signature generation and the public key for digital signature verification [ELGA84, ELGA85].

Before proceeding, we need a result from number theory. Recall from Chapter 2 that for a prime number $q$, if a is a primitive root of $q$, then

$$a, a^2, \subset, a^{q-1}$$

are distinct (mod $q$). It can be shown that, if a is a primitive root of $q$, then

1. For any integer $m$, $a^m$ K 1 (mod $q$) if and only if $m$ K 0 (mod $q$ - 1).
2. For any integers, $i, j$, $a^i$ K $a^j$ (mod $q$) if and only if $i$ K $j$ (mod $q$ - 1).

As with Elgamal encryption, the global elements of **Elgamal digital signature** are a prime number $q$ and a, which is a primitive root of $q$. User A generates a

private/public key pair as follows.

1. Generate a random integer $X_A$, such that $1 \leq X_A \leq q - 1$.
2. Compute $Y_A = a^{X_A} \bmod q$.
3. A's private key is $X_A$; A's pubic key is $\{q, a, Y_A\}$.

To sign a message $M$, user A first computes the hash $m = H(M)$, such that $m$ is an integer in the range $0 \ldots m \ldots q - 1$. A then forms a digital signature as follows.

1. Choose a random integer $K$ such that $1 \ldots K \ldots q - 1$ and $\gcd(K, q - 1) = 1$.That is, $K$ is relatively prime to $q - 1$.
2. Compute $S_1 = a^K \bmod q$. Note that this is the same as the computation of $C_1$ for Elgamal encryption.
3. Compute $K^{-1} \bmod (q - 1)$. That is, compute the inverse of $K$ modulo $q - 1$.
4. Compute $S_2 = K^{-1}(m - X_A S_1) \bmod (q - 1)$.
5. The signature consists of the pair $(S_1, S_2)$.

**SCHNORR DIGITAL SIGNATURE SCHEME**

As with the Elgamal digital signature scheme, the Schnorr signature scheme is based on discrete logarithms [SCHN89, SCHN91]. The Schnorr scheme minimizes the message-dependent amount of computation required to generate a signature. The main work for signature generation does not depend on the message and can be done during the idle time of the processor. The message-dependent part of the signature generation requires multiplying a $2n$-bit integer with an $n$-bit integer.

The scheme is based on using a prime modulus $p$, with $p - 1$ having a prime factor $q$ of appropriate size; that is, $p - 1 \equiv 0 \pmod q$. Typically, we use $p \approx 2^{1024}$ and $q \approx 2^{160}$. Thus, $p$ is a 1024-bit number, and $q$ is a 160-bit number, which is also
the length of the SHA-1 hash value.

The first part of this scheme is the generation of a private/public key pair,which consists of the following steps.

1. Choose primes $p$ and $q$, such that $q$ is a prime factor of $p - 1$.
2. Choose an integer $a$, such that $a^q = 1 \bmod p$. The values $a$, $p$, and $q$ comprise aglobal public key that can be common to a group of users.
3. Choose a random integer $s$ with $0 \leq s \leq q$. This is the user's private key.
4. Calculate $v = a^{-s} \bmod p$. This is the user's public key.

**NIST DIGITAL SIGNATURE ALGORITHM**

The National Institute of Standards and Technology (NIST) has published Federal Information Processing Standard FIPS 186, known as the Digital Signature Algorithm (DSA). The DSA makes use of the Secure Hash Algorithm (SHA). The DSA was originally proposed in 1991 and revised in 1993 in response to public feedback concerning the security of the scheme. There was a further minor revision in 1996. In 2000, an expanded version of the standard was issued as FIPS 186-2, subsequently updated to FIPS 186-3 in 2009, and FIPS 186-4 in 2013. This latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography.

## ➢ Authentication Protocols

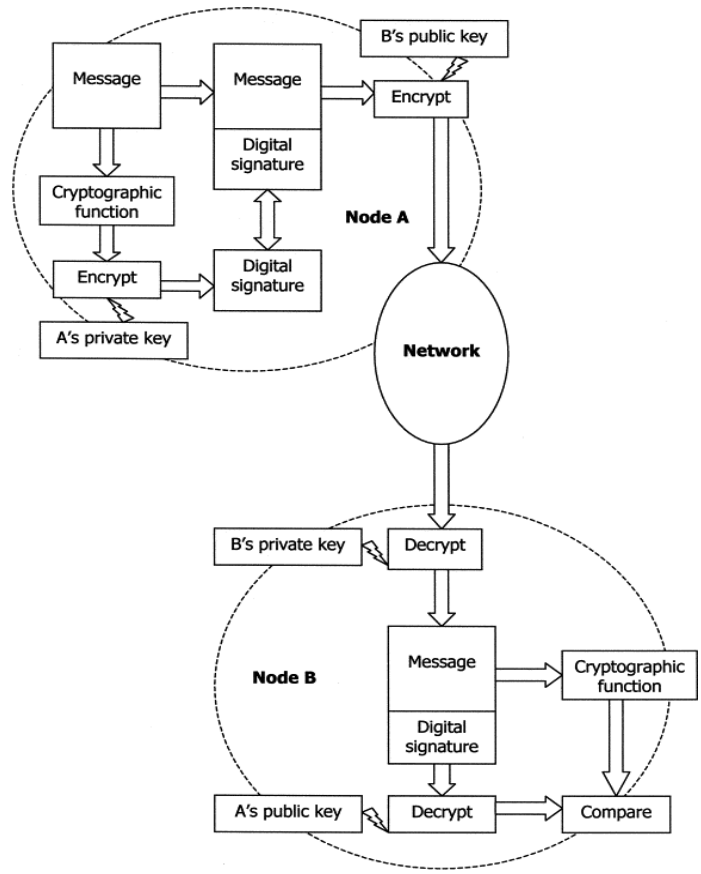
Authentication protocols can be based on shared secret key, public key, key distribution center, or the Kerberos protocol. The protocol based on shared secret key requires users A and B to share a secret key in order to use the protocol. The protocol consists of five message exchanges. A first sends a communication initiation message to B. B does not know whether this message is from A or from an intruder, so B sends a very large random number to A. To prove its identity, A then will encrypt the number with the shared secret key and return it to B. B will decrypt the message to obtain the original number back. Because only A and B know the secret key, B now knows that the message is coming from

A. Next, A sends a challenge (large random number) to B, B encrypts the number with the secret key and sends it back to A, and A decrypts the number to find out whether the message actually came from B. After this, the real communication can start. A problem with the secret key authentication is the secure distribution of the secret key.

The public key authentication protocol uses two keys per node, a public key for encryption and a private key for decryption. Everybody has access to the public key of a node, while the private key is secret. During authentication, random numbers are generated and exchanged, similar to the shared secret key protocol. The only difference is that the public key of the receiving node is used by the sending node to encrypt the random number, while the secret key of the receiving node is used to decrypt the received number. A disadvantage of this protocol is the non-trivial distribution of the public keys.

Another authentication method uses trusted key distribution centers (KDC). Each user has only one key that is shared with the distribution center. Whenever A wants to communicate with B, it generates a session key, encrypts the key with its own secret key, and sends it to the distribution center. The center knows A's secret key and is able to decrypt the session key. It then encrypts the session key with B's secret key and sends it to B, which is able to decrypt the session key again. The session key is then used for secure communication between A and B. To avoid replay attacks where intruders copy messages and resend them at a later time, time stamps or unique numbers are included in the messages to detect the message resending.

The Kerberos authentication protocol consists of a set of two additional servers, the authentication server (AS) and the ticket-granting server (TGS). The AS is similar to a key distribution center in that it shares a secret key with each user. To start a communication between users A and B, A contacts the AS. The server will send back a session key KS and a TGS ticket to A, both encrypted with A's secret key. The TGS ticket can later be used as proof that the sender is really A whenever A requests another ticket from the TGS server. A sends the request to communicate with B to the TGS. This request contains KS and the TGS ticket that A received from the AS. This ticket was encrypted by the AS using a secret TGS key. By decrypting the ticket, the TGS is therefore able to validate that it is communicating with user A. The TGS then creates a session key KAB for the A/B communication and sends two versions back to A: one version encrypted with KS and one version encrypted with B's secret key. A decrypts the first version by using its session key to obtain KAB. It then sends the second version to B, which is also able to decrypt KAB. Now A and B both have the same secret session key KAB to start a secure communication.




Authentication protocols authenticate users only. In many applications, such as financial transactions or e-commerce, messages themselves have to be authenticated as well. Digital signatures were therefore introduced. Digital signatures are used to verify the identity of the sender, to protect against

repudiation of the message by the sender later on, and to detect if a receiver has concocted a message himself. In general, any public key authentication algorithm can be used to produce digital signatures. For example, if user A wants to send a message with a signature to user B, A first generates the signature (by applying a cryptographic function such as a hash function to the message) and encrypts it with his private key and then with B's public key as shown in Fig. 5. After receiving the message and the signature, user B will decrypt the signature first with his private key and then with A's public key. After decrypting the actual message, B can generate a message signature and can compare it with the decrypted signature to verify the message and its sender.

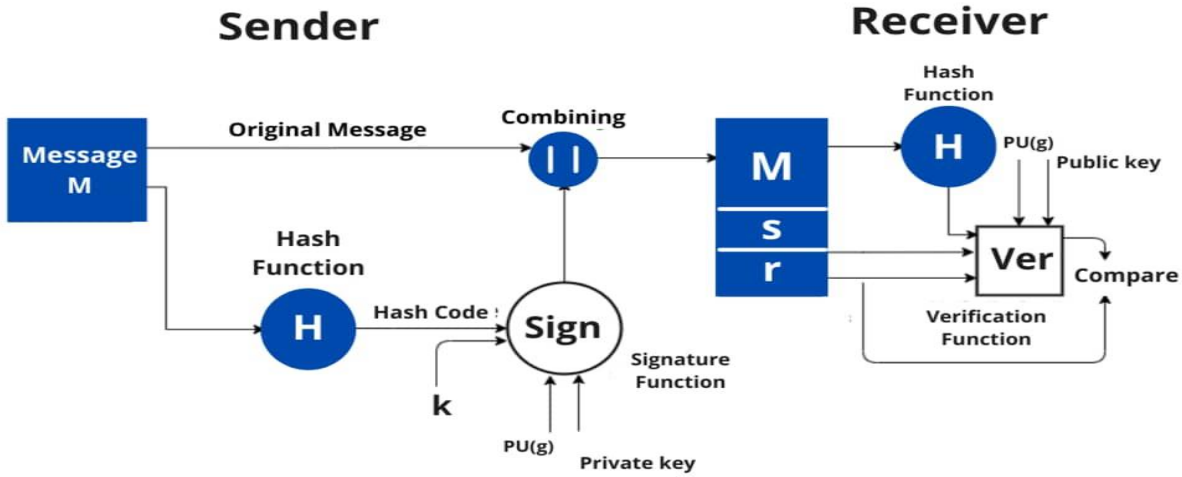
## ➢ DSS or Digital Signature Standard

It was introduced by the **National Institute of Standards and Technology** (NIST) in 1994. It has become the United States government standard for electronic document authentication. The Federal Information Processing Standard (FIPS) 186 specifies the DSS. It was first proposed in 1991 and revised in 1993as a result of public concerns about the scheme's security.

DSS employs SHA (**Secure Hash Algorithm)** to create digital signatures and offers a new digital signature mechanism known as the Digital Signature Algorithm.

What is the DSS approach?

The DSS is different from the fact that the RSA algorithm uses the public key, private key and hash function whereas the DSS uses the public key, private key, hash function, a random number k, and global public key. Therefore, DSS provides more security than RSA algorithms.




A hash code is generated from the message and given as an input to the signature function on the sender side. The other inputs to a signature function include a unique random number k for the signature, the private key of sender PR(a), and global public key i.e., PU(g).

The output of the signature function consists of two components: s & r, which is concatenated with the input message and then sent to the receiver.

Signature = {s, r}.

At the receiver side, the hash code for the message sent is generated by the receiver by applying a hash function. The verification function is used for verifying the message and signature sent by the sender. The verification function takes hash code generated, signature components s and r, the public key of the sender (PU(a)), and global public key

The signature function is compared with the output of the verification function and if both the values match, the signature is valid because A valid signature can only be generated by the sender using its private key.

Let's see what are the steps involved in DSS.
What are the steps involved in DSS?

- Generation of keys i.e., public and private keys for the source.
- Creation of digital signature by the source for a message.
- The receiver verifies the digital signature.

The DSA (Digital Signature Algorithm)
1. Generation of a global public key component

- Find a prime number p such that 2^L-1 < p < 2^L, where L is an integer between 512 and 1024 i.e., 512 <= L <= 1024.
- Find another number q which is a prime divisor of (p-1).
- Compute: g = h^(p-1)/q mod p, where h is an integer between 1 < h < p-1 and g should greater than 1 or h^(p-1)/q mod p>1.

2. Finding the user private and public keys

- The private key, x is any random number such that 0 < x< q.
- The public key, y = g^x mod p

3. Generating the Signature

- Finding the components of signature s and r.
- r = (gk mod p) mod q
- s = [k^ -1 {H(M) + x*r}] mod q where k is an integer such that 0 < k <q.
- Signature= (r, s)

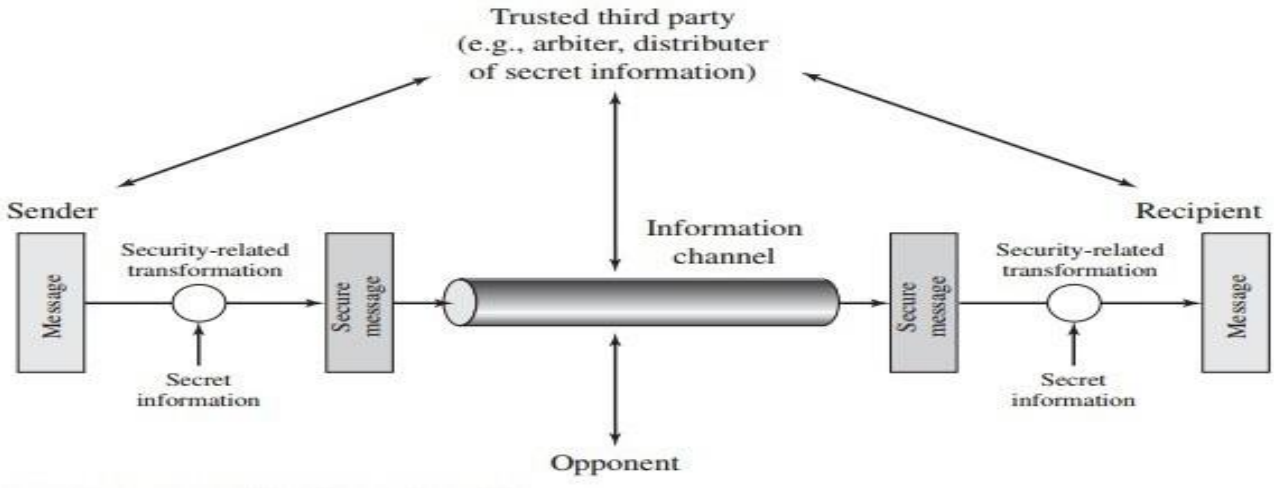
4. Verifying the signature
Let M', r', and s' be the message and signature components respectively received corresponding to M, r, and s.
To verify the signature is valid or not, first the following condition needs to be checked:
0 < r' < q and 0 < s' < q
If any of the above conditions is not met, the signature is considered invalid and is therefore discarded; otherwise, if both conditions are met, the following parameters are computed:
v = [(g^u1 y^u2) mod p] mod q
u1 = [H(M)w] mod q
u2 = (r') w mod q
w = (s')^ -1 mod q
Test: v = r'
If v = r', the signature is correct; otherwise, the data or message has been altered.
I hope this module will be useful and you now have a good knowledge of what is **Digital Signature Standard** (DSS) is in Cryptography. Stay tuned for more instructive and engaging modules like this.

## ➢ Network Security Practice

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

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

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the




authenticity of a message transmission.

This general model shows that there are four basic tasks in designing a particular securityservice:

**1.** Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.

**2.** Generate the secret information to be used with the algorithm.

**3.** Develop methods for the distribution and sharing of the secret information.

**4.** Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

Parts One through Five of this book concentrate on the types of security mechanisms and services that fit into the model shown in Figure. However, there are other security-related situations of interest that do not neatly fit this model but are considered in this book. A general model of these other situations is illustrated by Figure 1.5, which reflects a concern for protecting an information system from unwanted access. Most readers are familiar with the concerns caused by the existence of hackers, who attempt to

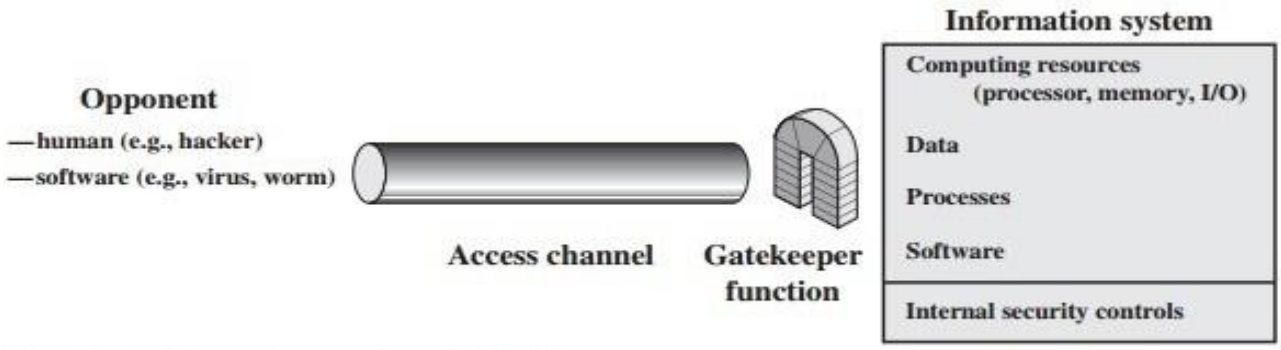



Figure 1.5 Network Access Security Model

penetrate systems that can be accessed over a network. The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system. The intruder can be a disgruntled employee who wishes to do damage or a criminal who seeks to exploit computer assets for financial gain

Another type of unwanted access is the placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application pro-grams as well as utility programs,such as editors and compilers. Programs can present two kinds of threats:

- **Information access threats:** Intercept or modify data on behalf of users who shouldnot have access to that data.
- **Service threats:** Exploit service flaws in computers to inhibit use by legitimate users.

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

**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 ofproviding security. Some of the mechanisms are

1 **Encipherment**
2 **Digital Signature**
3 **Access Control**

**SECURITY SERVIICES**

The classification of security services are as follows:

**Confidentiality:** Ensures that the information in a computer system and transmitted information are accessible only for reading by authorized parties.
E.g. Printing, displaying and other forms of disclosure.

**Authentication:** Ensures that the origin of a message or electronic document is correctly identified, with an assurance that the identity is not false.

**Integrity:** Ensures that only authorized parties are able to modify computer system assets and transmitted information. Modification includes writing, changing status, deleting, creating and delaying or replaying of transmitted messages.

**Non repudiation**: Requires that neither the sender nor the receiver of a message be able to deny the transmission.

**Access control**: Requires that access to information resources may be controlled by or the targetsystem.

**Availability**: Requires that computer system assets be available to authorized parties when needed.

## ➢ Authentication Protocol

An **authentication protocol** is a type of computer communications protocol or cryptographic protocol specifically designed for transfer of authentication data between two entities. It allows the receiving entity to authenticate the connecting entity (e.g. Client connecting to a Server) as well as authenticate itself to the connecting entity (Server to a client) by declaring the type of information needed for authentication as well as syntax. It is the most important layer of protection needed for secure communication within computer networks.

**Purpose**

With the increasing amount of trustworthy information being accessible over the network, the need for keeping unauthorized persons from access to this data emerged. Stealing someone's identity is easy in the computing world - special verification methods had to be invented to find out whether the person/computer requesting data is really who he says he is. The task of the authentication protocol is to specify the exact series of steps needed for execution of the authentication. It has to comply with the main protocol principles:

1. A Protocol has to involve two or more parties and everyone involved in the protocol must know the protocol in advance.
2. All the included parties have to follow the protocol.
3. A protocol has to be unambiguous - each step must be defined precisely.
4. A protocol must be complete - must include a specified action for every possible situation.

An illustration of password-based authentication using simple authentication protocol:
Alice (an entity wishing to be verified) and Bob (an entity verifying Alice's identity) are both aware of the protocol they agreed on using. Bob has Alice's password stored in a database for comparison.

1. Alice sends Bob her password in a packet complying with the protocol rules.
2. Bob checks the received password against the one stored in his database. Then he sends a packet saying "Authentication successful" or "Authentication failed" based on the result.

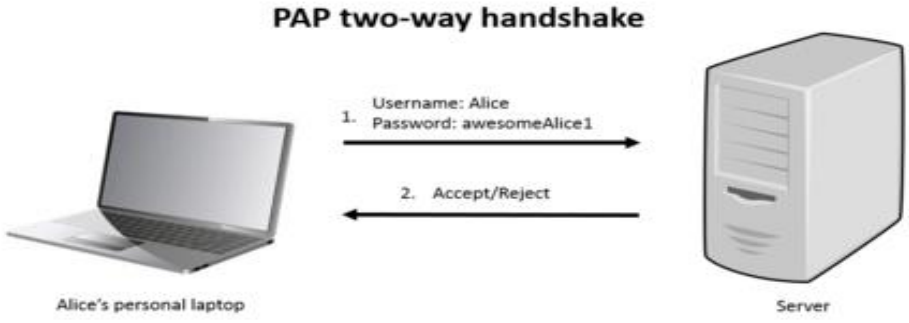
This is an example of a very basic authentication protocol vulnerable to many threats such as eavesdropping, replay attack, man-in-the-middle attacks, dictionary attacks or brute-force attacks. Most authentication protocols are more complicated in order to be resilient against these attacks.
Types

**Authentication protocols developed for PPP Point-to-Point Protocol**

Protocols are used mainly by Point-to-Point Protocol (PPP) servers to validate the identity of remote clients before granting them access to server data. Most of them use a password as the cornerstone of the authentication. In most cases, the password has to be shared between the communicating entities in advance.

**PAP - Password Authentication Protocol**

Password Authentication Protocol is one of the oldest authentication protocols. Authentication is initialized by the client sending a packet with credentials (username and password) at the beginning of the connection, with the client repeating the authentication request until acknowledgement is received. It is highly insecure because credentials are sent "in the clear" and repeatedly, making it vulnerable even to the most simple attacks like eavesdropping and man-in-the-middle based attacks. Although widely supported, it is specified that if an implementation offers a stronger authentication method, that method must be offered before PAP. Mixed authentication (e.g. the same client alternately using both PAP and CHAP) is also not expected, as the CHAP authentication would be compromised by PAP sending the password in plain-text.




**CHAP - Challenge-handshake authentication protocol**

The authentication process in this protocol is always initialized by the server/host and can be performed anytime during the session, even repeatedly. Server sends a random string (usually 128B long). The client uses password and the string received as parameters for MD5 hash function and then sends the result together with username in plain text. Server uses the username to apply the same function and compares the calculated and received hash. An authentication is successful or unsuccessful.

**EAP - Extensible Authentication Protocol**

EAP was originally developed for PPP(Point-to-Point Protocol) but today is widely used in IEEE 802.3, IEEE 802.11(WiFi) or IEEE 802.16 as a part of IEEE 802.1x authentication framework. The latest version is standardized in RFC 5247. The advantage of EAP is that it is only a general authentication framework for client-server authentication - the specific way of authentication is defined in its many versions called EAP-methods. More than 40 EAP-methods exist, the most common are:

- EAP-MD5
- EAP-TLS
- EAP-TTLS
- EAP-FAST
- EAP-PEAP

**AAA architecture protocols (Authentication, Authorization, Accounting)**

Complex protocols used in larger networks for verifying the user (Authentication), controlling access to server data (Authorization) and monitoring network resources and information needed for billing of services (Accounting).

**TACACS, XTACACS and TACACS+**

The oldest AAA protocol using IP based authentication without any encryption (usernames and passwords were transported as plain text). Later version XTACACS (Extended TACACS) added authorization and accounting. Both of these protocols were later replaced by TACACS+. TACACS+ separates the AAA components thus they can be segregated and handled on separate servers (It can even use another protocol for e.g. Authorization). It uses TCP (Transmission Control Protocol) for transport and encrypts the whole packet. TACACS+ is Cisco proprietary.

**RADIUS**

Remote Authentication Dial-In User Service (RADIUS) is a full AAA protocol commonly used by ISP. Credentials are mostly username-password combination based, it uses NAS and UDP protocol for transport.

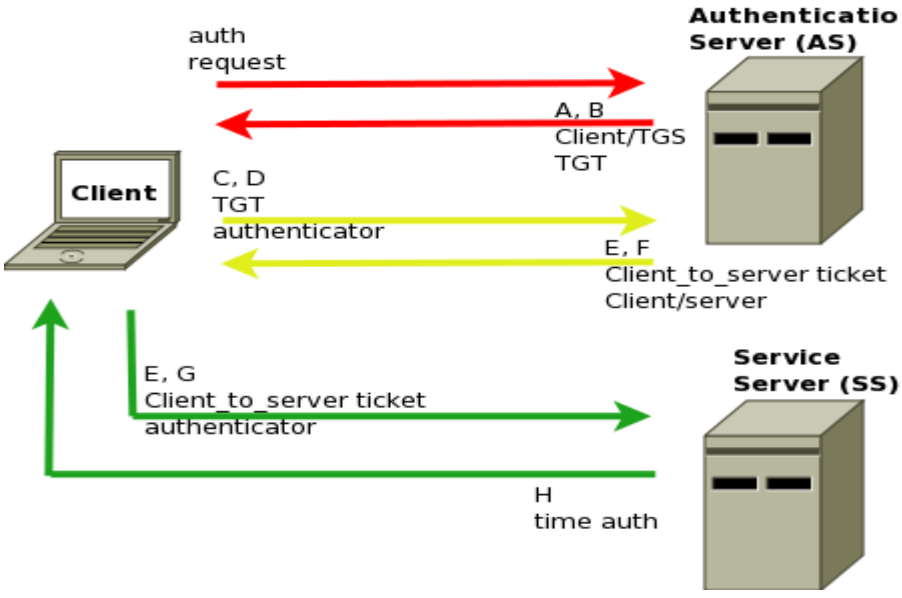
**DIAMETER**

Diameter (protocol) evolved from RADIUS and involves many improvements such as usage of more reliable TCP or SCTP transport protocol and higher security thanks to TLS.

**Other**

**Kerberos (protocol)**

Kerberos is a centralized network authentication system developed at MIT and available as a free implementation from MIT but also in many commercial products. It is the default authentication method in Windows 2000 and later. The authentication process itself is much more complicated than in the previous protocols - Kerberos uses symmetric key cryptography, requires a trusted third party and can use public-key cryptography during certain phases of authentication if need be.




## ➢ Basic overview of Electronic Mail Security:

In virtually all distributed environments, electronic mail is the most heavily used network-based application. Users expect to be able to, and do, send e-mail to others who are connected directly or indirectly to the Internet, regardless of host operating system or communications suite. With the explosively growing reliance on e-mail, there grows a demand for authentication and confidentiality services. Two schemes stand out as approaches that enjoy widespread use: Pretty Good Privacy (PGP) and S/MIME. Both are examined in this chapter. The chapter closes with a discussion of Domain Keys Identified Mail.

## ➢ PRETTY GOOD PRIVACY

PGP is a remarkable phenomenon. Largely the effort of a single person, Phil Zimmermann, PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. In essence, Zimmermann has done the following:

1. Selected the best available cryptographic algorithms as building blocks.
2. Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands.
3. Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks such as AOL (America On Line).
4. Entered into an agreement with a company (Via crypt, now Network Associates) to provide a fully compatible, low-cost commercial version of PGP. PGP has grown explosively and is now widely used.

The PGP documentation often uses the term *secret key* to refer to a key paired with a public key in a public-key encryption scheme. As was mentioned earlier, this practice risks confusion with a secret key used for symmetric encryption. Hence, we use the term *private key* instead.

**Operational Description**

The actual operation of PGP, as opposed to the management of keys, consists of four services: authentication, confidentiality, compression, and e-mail compatibility

**AUTHENTICATION** Figure illustrates the digital signature service provided by PGP.

.The sequence is as follows.

**1.** The sender creates a message.

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

CONFIDENTIALITY another basic service provided by PGP is confidentiality, which is provided by encrypting messages to be transmitted or to be stored locally as files. In both cases, the symmetric encryption algorithm CAST-128 may be used. Alternatively, IDEA or 3DES may be used. The 64-bit cipher feedback (CFB) mode is used. As always, one must address the problem of key distribution. In PGP, each symmetric key is used only once. That is, a new key is generated as a random 128- bit number for each message. Thus, although this is referred to in the documentation as a session key, it is in reality a one-time key. Because it is to be used only once, the session key is bound to the message and transmitted with it.To protect the key, it is encrypted with the receiver's public key. Figure illustrates the sequence, which canbe described as follows.

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

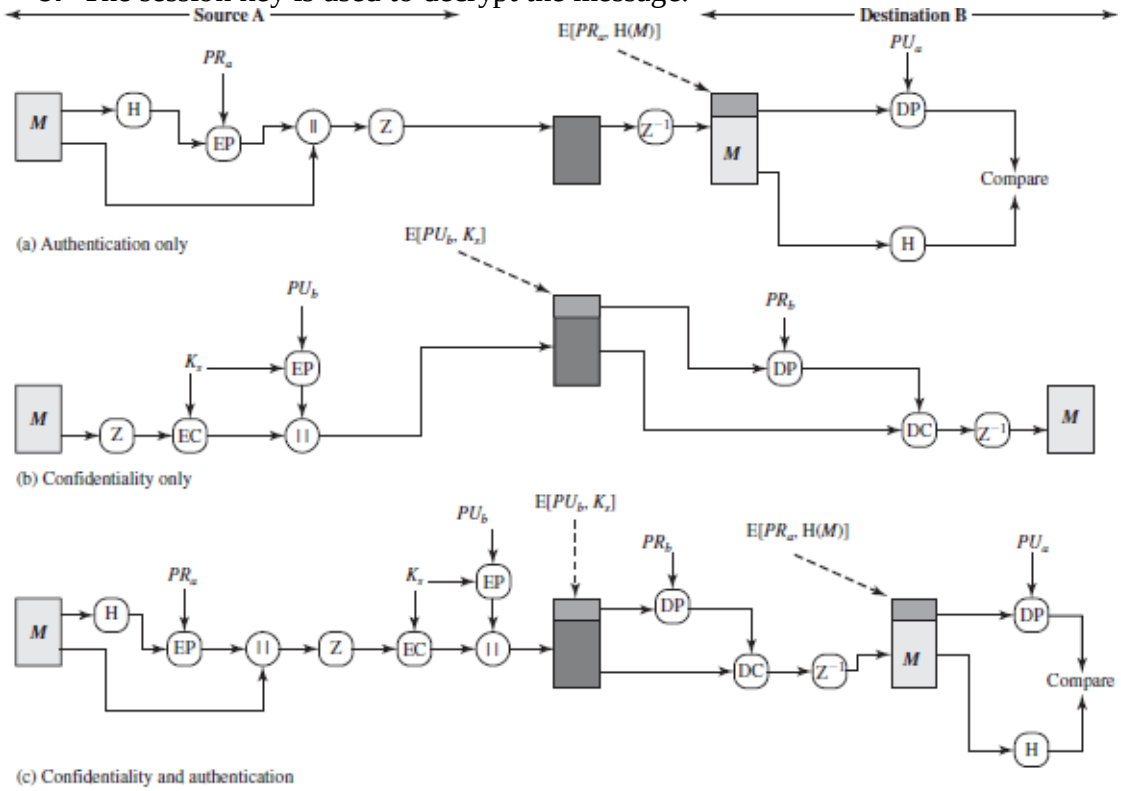
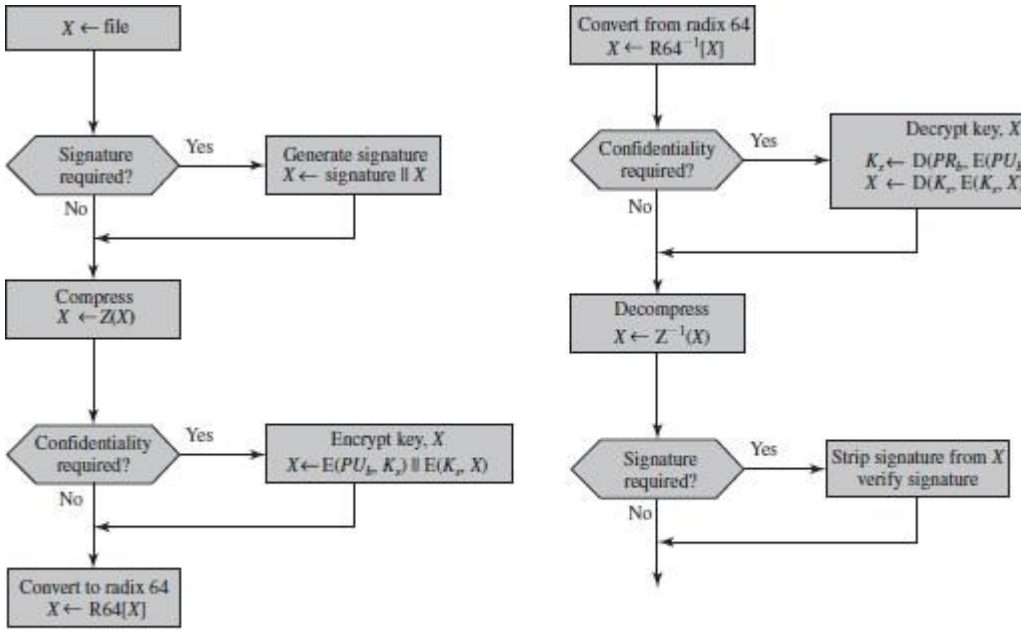



Figure PGP Cryptographic Functions




(a) Generic transmission diagram (from A)

(b) Generic reception diagram (to B)

Figure Transmission and Reception of PGP Messages

The **message component** includes the actual data to be stored or transmitted, as well as a filename and a timestamp that specifies the time of creation. The **signaturecomponent** includes the following.

- **Timestamp:** The time at which the signature was made.
- **Message digest:** The 160-bit SHA-1 digest encrypted with the sender's private signature key. The digest is calculated over the signature timestamp concatenated with the data portion of the message component. The inclusion of the signature timestamp in the digest insures against replay types of attacks. The exclusion of the filename and timestamp portions of the message component ensures that detached signatures are exactly the same as attached signatures prefixed to the message. Detached signatures are calculated on a separate file that has none of the message component header fields.
- **Leading two octets of message digest:** Enables the recipient to determine if the correct public key was used to decrypt the message digest for authentication by comparing this plaintext copy of the first two octets with the first two octets of the decrypted digest. These octets also serve as a 16-bit frame check sequence for the message.
- **Key ID of sender's public key:** Identifies the public key that should be used to decrypt the message digest and, hence, identifies the private key that was used to encrypt the message digest.

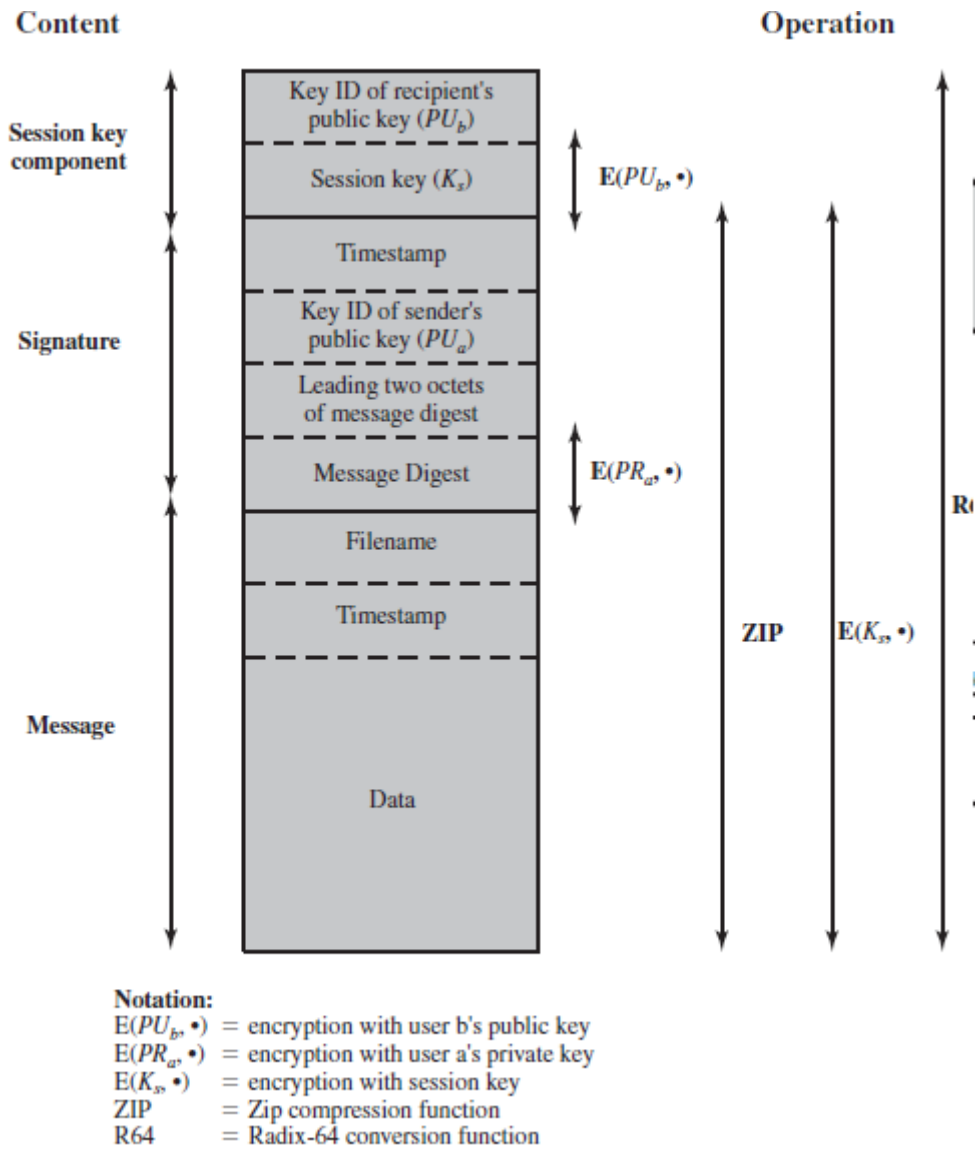



Figure General Format PGP Message (from A to B)

## ➢ S/MIME

Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security. Although both PGP and S/MIME are on an IETF standards track, it appears likely that S/MIME will emerge as the industry standard for commercial and organizational use, while PGP will remain the choice for personal e-mail security for many users.

**Multipurpose Internet Mail Extensions**

Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework that is intended to address some of the problems and limitations of the use of Simple Mail Transfer Protocol (SMTP), defined in RFC 821, or some other mail transfer protocol and RFC 5322 for electronic mail. [PARZ06] lists the following limitations of the SMTP/5322 scheme.

1. SMTP cannot transmit executable files or other binary objects. A number of schemes are in use for converting binary files into a text form that can be used by SMTP mail systems, including the popular UNIX UUencode/Uudecode scheme.However, none of these is a standard or even a *de facto* standard.

**2.** SMTP cannot transmit text data that includes national language characters, because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.

**3.** SMTP servers may reject mail message over a certain size.

**4.** SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.

**5.** SMTP gateways to X.400 electronic mail networks cannot handle non textual data included in X.400 messages.

**6.** Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include:

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

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 5322 implementations. The specification is provided in RFCs 2045 through 2049.

**OVERVIEW** The MIME specification includes the following elements.

**1.** Five new message header fields are defined, which may be included in an RFC 5322 header. These fields provide information about the body of the message.

**2.** A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.

**3.** Transfer encodings are defined that enable the conversion of any content format into a form that is protected from alteration by the mail system.

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

The five header fields defined in MIME are

- **MIME-Version:** Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- **Content-Type:** Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
- **Content-Transfer-Encoding:** Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.
- **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
- **Content-Description:** A text description of the object with the body; this is useful when the object is not readable (e.g., audio data).

**S/MIME Messages**

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

We examine each of these in turn after first looking at the general procedures for S/MIME message preparation.

**SECURING A MIME ENTITY** S/MIME secures a MIME entity with a signature, encryption, or both. A MIME entity may be an entire message (except for the RFC 5322 headers), or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message. The MIME entity is prepared according to the normal rules for MIME message preparation. Then the MIME entity plus some security-related data, such as algorithm identifiers and certificates, are processed by S/MIME to produce

what is known as a PKCS object. A PKCS object is then treated as message content and wrapped in MIME (provided with appropriate MIME headers). This process should become clear as we look at specific objects and provide examples.

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

The use of transfer encoding requires special attention. For most cases, the result of applying the security algorithm will be to produce an object that is partially or totally represented in arbitrary binary data. This will then be wrapped in an outer MIME message, and transfer encoding can be applied at that point, typically base64. However, in the case of a multipart signed message (described in more detail later), the message content in one of the subparts is unchanged by the security process. Unless that content is 7bit, it should be transfer encoded using base 64 or quoted- printable so that there is no danger of altering the content to which the signature wasapplied.

## ➢ IP SECURITY

IPsec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include:

- **Secure branch office connectivity over the Internet:** A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead.
- **Secure remote access over the Internet:** An end user whose system is equipped with IP security protocols can make a local call to an Internet Service Provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for traveling employees and telecommuters.
- **Establishing extranet and intranet connectivity with partners:** IPSec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.
- **Enhancing electronic commerce security:** Even though some Web and electronic commerce applications have built-in security protocols, the use of IPSec enhances that security. IPsec guarantees that all traffic designated by the network administrator is both encrypted and authenticated, adding an additional layer of security to whatever is provided at the application layer.

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

Figure is a typical scenario of IPsec usage. An organization maintains LANs at dispersed locations. Non secure IP traffic is conducted on each LAN. For traffic offsite, through some sort of private or public WAN, IPsec protocols are used.

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

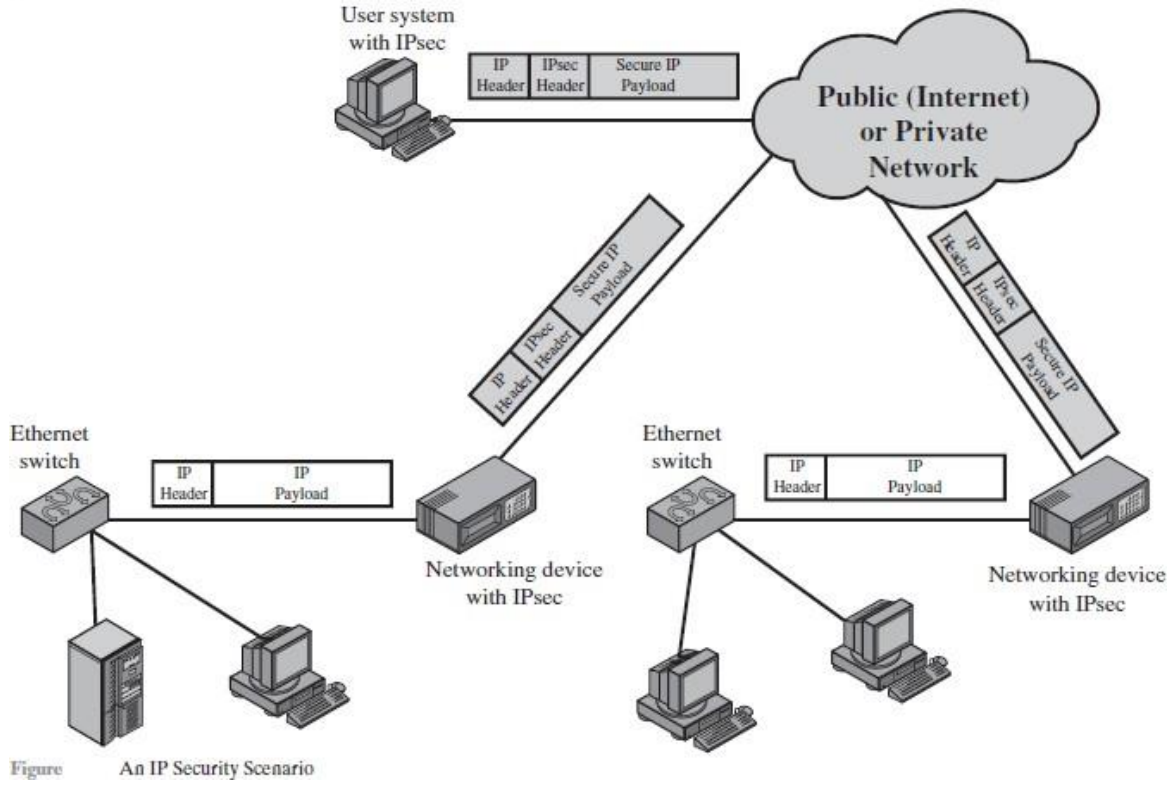



Figure An IP Security Scenario

**Benefits of IPsec**

Some of the benefits of IPsec are:

• When IPsec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.

• IPsec in a firewall is resistant to bypass if all traffic from the outside must use IP and the firewall is the only means of entrance from the Internet into the organization.

• IPsec is below the transport layer (TCP, UDP) and so is transparent to applications. There is no need to change software on a user or server system when IPsec is implemented in the firewall or router. Even if IPsec is implemented in end systems, upper-layer software, including applications, is not affected.

• IPsec can be transparent to end users. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization.

• IPsec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual sub network within an organization for sensitive applications.

**Routing Applications**

In addition to supporting end users and protecting premises systems and networks, IPsec can play a vital role in the routing architecture required for internetworking. The following are the examples of the use of IPsec. IPsec can assure that

• A router advertisement (a new router advertises its presence) comes from an authorized router.

• A neighbor advertisement (a router seeks to establish or maintain a neighbour relationship with a router in another routing domain) comes from an authorized router.

• A redirect message comes from the router to which the initial IP packet was sent.

• A routing update is not forged.

Without such security measures, an opponent can disrupt communications or divert some traffic. Routing protocols such as Open Shortest Path First (OSPF) should be run on top of security associations between

routers that are defined by IPsec.

**IPsec Documents**

IPsec encompasses three functional areas: authentication, confidentiality, and key management. The totality of the IPsec specification is scattered across dozens of RFCs and draft IETF documents, making this the most complex and difficult to grasp of all IETF specifications.

The documents can be categorized into the following groups.

- **Architecture:** Covers the general concepts, security requirements, definitions, and mechanisms defining IPsec technology. The current specification is RFC 4301, Security Architecture for the Internet Protocol.
- **Authentication Header (AH):** AH is an extension header to provide message authentication. The current specification is RFC 4302, IP Authentication Header. Because message authentication is provided by ESP, the use of AH is deprecated. It is included in IPsecv3 for backward compatibility but should not be used in new applications.
- **Encapsulating Security Payload (ESP):** ESP consists of an encapsulating header and trailer used to provide encryption or combined encryption/authentication. The current specification is RFC 4303, IP Encapsulating Security Payload (ESP).
- **Internet Key Exchange (IKE):** This is a collection of documents describing the key management schemes for use with IPsec. The main specification is RFC 4306, Internet Key Exchange (IKEv2) Protocol, but there are a number of related RFCs.
- **Cryptographic algorithms:** This category encompasses a large set of documents that define and describe cryptographic algorithms for encryption, message authentication, pseudorandom functions (PRFs), and cryptographic key exchange.
- **Other:** There are a variety of other IPsec-related RFCs, including those dealing with security policy and management information base (MIB) content.

**IPsec Services**

IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services. Two protocols are used to provide security: an authentication protocol designated by the header of the protocol,Authentication Header (AH); and a combined encryption/ authentication protocol designated by the format of the packet for that protocol, Encapsulating Security Payload (ESP). RFC 4301 lists the following services:

- Access control
- Connectionless integrity
- Data origin authentication
- Rejection of replayed packets (a form of partial sequence integrity)
- Confidentiality (encryption)
- Limited traffic flow confidentiality

**Transport and Tunnel Modes**

Both AH and ESP support two modes of use: transport and tunnel mode. The operation of these two modes is best understood in the context of a description of ESP, which is covered in Section 19.3. Here we provide a brief overview.

**TRANSPORT MODE** Transport mode provides protection primarily for upper-layer protocols. That is, transport mode protection extends to the payload of an IP packet.1 Examples include a TCP or UDP segment or an ICMP packet, all of which operate directly above IP in a host protocol stack. Typically, transport mode is used for end-to-end communication between two hosts (e.g., a client and a server, or two workstations).When a host runs AH or ESP over IPv4, the payload is the data that normally follow the IP header. For IPv6, the payload is the data that normally follow both the IP header and any IPv6

extensions headers that are present, with the possible exception of the destination options header, which may be included in the protection.

ESP in transport mode encrypts and optionally authenticates the IP payload but not the IP header. AH in transport mode authenticates the IP payload andselected portions of the IP header.

**TUNNEL MODE** Tunnel mode provides protection to the entire IP packet. To achieve this, after the AH or ESP fields are added to the IP packet, the entire packet plus security fields is treated as the payload of new outer IP packet with a new outer IP header. The entire original, inner, packet travels through a tunnel from one point of an IP network to another; no routers along the way are able to

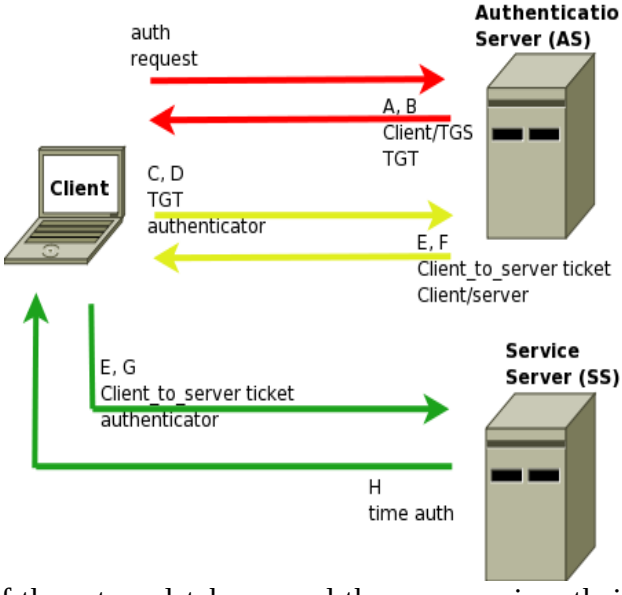
examine the inner IP header. Because the original packet is encapsulated, the new, larger packet may have totally different source and destination addresses, adding

to the security.Tunnel mode is used when one or both ends of a security association (SA) are a security gateway, such as a firewall or router that implements IPsec.

With tunnel mode, a number of hosts on networks behind firewalls may engage in secure communications without implementing IPsec. The unprotected packets generated by such hosts are tunneled through external networks by tunnel mode SAs set up by the IPsec software in the firewall or secure router at the boundary ofthe local network.

Fundamental to the operation of IPsec is the concept of a security policy applied to each IP packet that transits from a source to a destination. IPsec policy is

determined primarily by the interaction of two databases, the **security association database (SAD)** and the **security policy database (SPD)**. This section provides an overview o



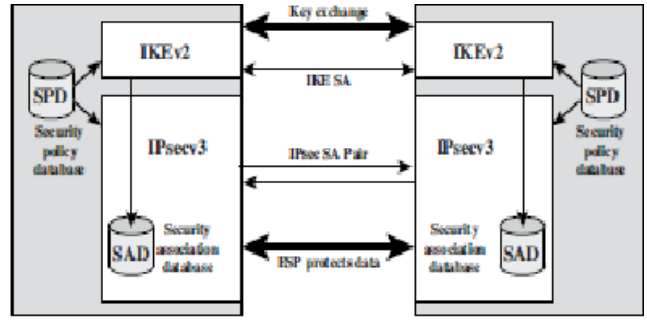




Figure IPsec Architecture

f these two databases and then summarizes their use during IPsec operation. Figure 19.2 illustrates the relevant relationships.

## ➢ WEB SECURITY

### SSL Architecture

SSL is designed to make use of TCP to provide a reliable end-to-end secure service. SSL is not a single protocol but rather two layers of protocols, as illustrated in Figure. The SSL Record Protocol provides basic security services to various higher layer protocols. In particular, the Hypertext Transfer Protocol (HTTP), which provides the transfer service for Web client/server interaction, can operate on top of SSL. Three higher-layer protocols are defined as part of SSL: the Handshake Protocol, The Change Cipher Spec Protocol, and the Alert Protocol. These SSL-specific protocols are used in the management of SSL

exchange.

Two important SSL concepts are the SSL session and the SSL connection, which are defined in the specification as follows.

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

• **Session:** An SSL session is an association between a client and a server. Sessions are created by the Handshake Protocol. Sessions define a set of cryptographic

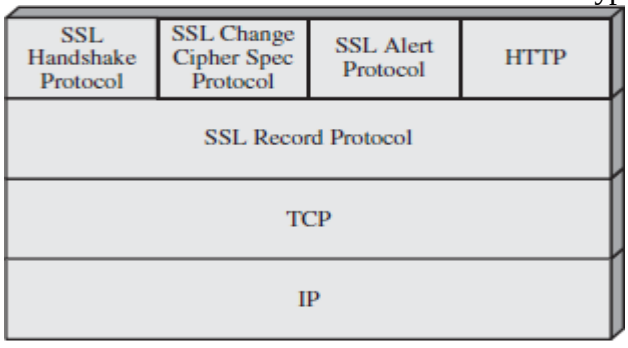



Figure SSL Protocol Stack

security parameters which can be shared among multiple connections. Sessions are used to avoid the expensive negotiation of new security parameters for each connection. Between any pair of parties (applications such as HTTP on client and server), there may be multiple secure connections. In theory, there may also be multiple simultaneous sessions between parties, but this feature is not used in practice. There are a number of states associated with each session. Once a session is established, there is a current operating state for both read and write (i.e., receive and send). In addition, during the Handshake Protocol, pending read and write states are created. Upon successful conclusion of the Handshake Protocol, the pending states become the current states.

A session state is defined by the following parameters.

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

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

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

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

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

• **Is resumable:** A flag indicating whether the session can be used to initiate new connections. A connection state is defined by the following parameters.

• **Server and client random:** Byte sequences that are chosen by the server and client for each connection.

• **Server write MAC secret:** The secret key used in MAC operations on data sent bythe server.

• **Client write MAC secret:** The secret key used in MAC operations on data sent bythe client.

• **Server write key:** The secret encryption key for data encrypted by the server and decrypted by the client.

• **Client write key:** The symmetric encryption key for data encrypted by the client and decrypted by the server.

• **Initialization vectors:** When a block cipher in CBC mode is used, an initialization vector (IV) is

maintained for each key. This field is first initialized by the SSL Handshake Protocol. Thereafter, the final ciphertext block from each record is preserved for use as the IV with the following record.

• **Sequence numbers:** Each party maintains separate sequence numbers for transmitted and received messages for each connection.When a party sends or receives a change cipher spec message, the appropriate sequence number is set to zero. Sequence numbers may not exceed 264 – 1.

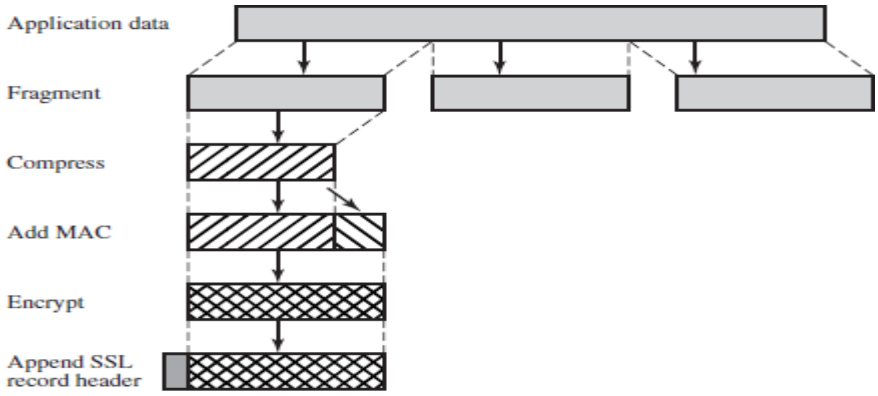



Figure SSL Record Protocol Operation

**SSL Record Protocol**

The SSL Record Protocol provides two services for SSL connections:

• **Confidentiality:** The Handshake Protocol defines a shared secret key that is used for conventional encryption of SSL payloads.

• **Message Integrity:** The Handshake Protocol also defines a shared secret key that is used to form a message authentication code (MAC).

Figure 16.3 indicates the overall operation of the SSL Record Protocol. The Record Protocol takes an application message to be transmitted, fragments the data into manageable blocks, optionally compresses the data, applies a MAC, encrypts, adds a header, and transmits the resulting unit in a TCP segment. Received data are decrypted, verified, decompressed, and reassembled before being delivered to higher-level users.

The first step is **fragmentation**. Each upper-layer message is fragmented into blocks of 214 bytes (16384 bytes) or less. Next, **compression** is optionally applied. Compression must be lossless and may not increase the content length by more than 1024 bytes.1In SSLv3 (as well as the current version of TLS), no compression algorithm is specified, so the default compression algorithm is null. The next step in processing is to compute a **message authentication code** over the compressed data. For this purpose, a shared secret key is used. The calculation is defined as hash(MAC_write_secret || pad_2|| hash(MAC_write_secret || pad_1||seq_num || SSLCompressed.type || SSLCompressed.length || SSLCompressed.fragment))

where

|| = concatenation

MAC_write_secret = shared secret key hash = cryptographic hash algorithm; eitherMD5 or SHA-1

pad_1 = the byte 0x36 (0011 0110) repeated48 times (384 bits) for MD5 and 40 times (320 bits) for SHA-1

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

seq_num = the sequence number for this message SSLCompressed.type = the higher-level protocol used to processthis fragment

SSLCompressed.length = the length of the compressed fragment SSLCompressed.fragment = the compressed fragment (if compressionis not used, this is the plaintext fragment)

## ➢ INTRUDERS

One of the two most publicized threats to security is the intruder (the other is viruses), often referred to as a hacker or cracker. In an important early study of intrusion, Anderson identified three classes of intruders:

- **Masquerader:** An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account
- **Misfeasor:** A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges
- **Clandestine user:** An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

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

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

- Performing a remote root compromise of an e-mail server
- Defacing a Web server
- Guessing and cracking passwords
- Copying a database containing credit card numbers
- Viewing sensitive data, including payroll records and medical information,without authorization
- Running a packet sniffer on a workstation to capture usernames and passwords
- Using a permission error on an anonymous FTP server to distribute piratedsoftware and music files
- Dialing into an unsecured modem and gaining internal network access
- Posing as an executive, calling the help desk, resetting the executive's e-mail password, and learning the new password
- Using an unattended, logged-in workstation without permission

**Some Examples of Intruder Patterns of Behavior**

**(a) Hacker**

1. Select the target using IP lookup tools such as NSLookup, Dig, and others.
2. Map network for accessible services using tools such as NMAP.
3. Identify potentially vulnerable services (in this case, pcAnywhere).
4. Brute force (guess) pcAnywhere password.
5. Install remote administration tool called DameWare.Wait for administrator to log on and capture his password.
6. Use that password to access remainder of network.

**(b) Criminal Enterprise**

1. Act quickly and precisely to make their activities harder to detect.
2. Exploit perimeter through vulnerable ports.
3. Use Trojan horses (hidden software) to leave back doors for reentry.
4. Use sniffers to capture passwords.
5. Do not stick around until noticed.
6. Make few or no mistakes.

**(c) Internal Threat**

1. Create network accounts for themselves and their friends.
2. Access accounts and applications they wouldn't normally use for their daily jobs.
3. E-mail former and prospective employers.

4. Conduct furtive instant-messaging chats.
5. Visit Web sites that cater to disgruntled employees, such as f'dcompany.com.
6. Perform large downloads and file copying.
7. Access the network during off hours.

**Intrusion Techniques**

The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. Most initial attacks use system or software vulnerabilities that allow a user to execute code that opens a back door into the system. Alternatively, the intruder attempts to acquire information that should have been protected. In some cases, this information is in the form of a user password. With knowledge of some other user's password, an intruder can log in to a system and exercise all the privileges accorded to the legitimate user. Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it and learn passwords. The password file can be protected in one of two ways:

- **One-way function:**

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

- **Access control:**

Access to the password file is limited to one or a very few accounts

**Few techniques for learning passwords:**

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

**Intrusion Detection System:**

An intrusion detection system (IDS) is a device, typically another separate computer that monitors activity to identify malicious or suspicious events. An IDS is a sensor, like a smoke detector, that raises an alarm if specific things occur. A model of an IDS is shown in below figure. The components in the figure are the four basic elements of an intrusion detection system, based on the Common Intrusion Detection Framework of [STA96]. An IDS receives raw inputs from sensors. It saves those inputs, analyzesthem, and takes some controlling action.

**Types of IDSs**

The two general types of intrusion detection systems are signature based and heuristic. Signature-based intrusion detection systems perform simple pattern- matching and report situations that match a pattern corresponding to a known attack type. Heuristic intrusion detection systems, also known as anomaly based, build a model of acceptable behavior and flag exceptions to that model; for the future, the administrator can mark a flagged behavior as acceptable so that the heuristic IDS will now treat that previously unclassified behavior as acceptable.

Intrusion detection devices can be network based or host based. A network-based IDS is a stand-alone

device attached to the network to monitor traffic throughout that network; a host-based IDS runs on a single workstation or client or host, to protect that one host.

**Signature-Based Intrusion Detection:**

A simple signature for a known attack type might describe a series of TCP SYN packets sent to many different ports in succession and at times close to one another, as would be the case for a port scan. An intrusion detection system would probably find nothing unusual in the first SYN, say, to port 80, and then another (from the same source address) to port 25. But as more and more ports receive SYN packets, especially ports that are not open, this pattern reflects a possible port scan. Similarly, some implementations of the protocol stack fail if they receive an ICMP packet with a data length of 65535 bytes, so such a packet would be a pattern for which to watch. *Heuristic Intrusion* **Detection:**

Because signatures are limited to specific, known attack patterns, another form of intrusion detection becomes useful. Instead of looking for matches, heuristic intrusion detection looks for behavior that is out of the ordinary. The original work in this area focused on the individual, trying to find characteristics of that person that might be helpful in understanding normal and abnormal behavior. For example, one user might always start the day by reading e-mail, write many documents using a word processor, and occasionally back up files. These actions would be normal. This user does not seem to use many administrator utilities. If that person tried to access sensitive system management utilities, this new behavior might be a clue that someone else was acting under the user's identity.

Inference engines work in two ways. Some, called state-based intrusion detection systems, see the system going through changes of overall state or configuration. They try to detect when the system has veered into unsafe modes. Others try to map current activity onto a model of unacceptable activity and raise an alarm when the activity resembles the model.

These are called model-based intrusion detection systems. This approach has been extended to networks in [MUK94]. Later work sought to build a dynamic model of behavior, to accommodate variation and evolution in a person's actions over time. The technique compares real activity with a known representation of normality.

Alternatively, intrusion detection can work from a model of known bad activity. For example, except for a few utilities (login, change password, create user), any other attempt to access a password file is suspect. This form of intrusion detection is known as misuse intrusion detection. In this work, the real activity is compared against a known suspicious area.

Stealth Mode:

An IDS is a network device (or, in the case of a host-based IDS, a program running on a network device). Any network device is potentially vulnerable to network attacks. How useful would an IDS be if it itself were deluged with a denial-of-service attack? If an attacker succeeded in logging in to a system within the protected network, wouldn't trying to disable the IDS be the next step?

To counter those problems, most IDSs run in stealth mode, whereby an IDS has two network interfaces: one for the network (or network segment) being monitored and the other to generate alerts and perhaps other administrative needs. The IDS uses the monitored interface as input only; it *never* sends packets out through that interface. Often, the interface is configured so that the device has no published address through the monitored interface; that is, a router cannot route anything to that address directly, because the router does not know such a device exists. It is the perfect passive wiretap. If the IDS needs to generate an alert, it uses only the alarm interface on a completely separate control network.

**Goals for Intrusion Detection Systems:**

*1.* Responding to alarms:

Whatever the type, an intrusion detection system raises an alarm when it finds a match. The alarm can range from something modest, such as writing a note in an audit log, to something significant, such as

paging the system security administrator. Particular implementations allow the user to determine what action the system should take on what events. In general, responses fall into three major categories (any or all of which can beused in a single response):

Monitor, collect data, perhaps increase amount of data collectedProtect, act to reduce exposure

Call a human

*2.* False Results:

Intrusion detection systems are not perfect, and mistakes are their biggest problem. Although an IDS might detect an intruder correctly most of the time, it may stumble in two different ways: by raising an alarm for something that is not really an attack (called a false positive, or type I error in the statistical community) or not raising an alarm for a real attack (a false negative, or type II error). Too many false positives means the administrator will be less confident of the IDS's warnings, perhaps leading to a real alarm's being ignored. But false negatives mean that real attacks are passing the IDS without action. We say that the degree of false positives and false negatives represents the sensitivity of the system. Most IDS implementations allow the administrator to tune the system's sensitivity, to strike an acceptable balance between false positives and negatives.

**IDS strength and limitations:**

On the upside, IDSs detect an ever-growing number of serious problems. And as we learn more about problems, we can add their signatures to the IDS model. Thus, over time, IDSs continue to improve. At the same time, they are becoming cheaper and easier to administer. On the downside, avoiding an IDS is a first priority for successful attackers. An IDS that is not well defended is useless. Fortunately, stealth mode IDSs are difficult even to find on an internal network, let alone to compromise. IDSs look for known weaknesses, whether through patterns of known attacks or models of normal behavior. Similar IDSs may have identical vulnerabilities, and their selection criteria may miss similar attacks. Knowing how to evade a particular model of IDS is an important piece of intelligence passed within the attacker community. Of course, once manufacturers become aware of a shortcoming in their products, they try to fix it. Fortunately, commercial IDSs are pretty good at identifying attacks. Another IDS limitation is its sensitivity, which is difficult to measure and adjust. IDSs will never be perfect, so finding the proper balance is critical.

In general, IDSs are excellent additions to a network's security. Firewalls block traffic to particular ports or addresses; they also constrain certain protocols to limit their impact. But by definition, firewalls have to allow some traffic to enter a protected area. Watching what that traffic actually does inside the protected area is an IDS's job, which it does quite well.

**Trojan Horses**

A Trojan horse is a useful, or apparently useful, program or command procedure containing hidden code that, when invoked, performs some unwanted or harmful function. Trojan horse programs can be used to accomplish functions indirectly that an unauthorized user could not accomplish directly. For example, to gain access to the files of another user on a shared system, a user could create a Trojan horse program that, when executed, changes the invoking user’s file permissions so that the files are readable by any user. The author could then induce users to run the program by placing it in a common directory and naming it such that it appears to be a useful utility program or application. An example is a program that ostensibly produces a listing of the user’s files in a desirable format. After another user has run the program, the author of the program can then access the information in the user’s files. An example of a Trojan horse program that would be difficult to detect is a compiler that has been modified to insert additional code into certain programs as they are compiled, such as a system login program. The code creates a backdoor in the login program that permits the author to log on to the system using a special password. This Trojan horse can never be discovered by reading the source code of the login program.

Trojan horses fit into one of three models:

- Continuing to perform the function of the original program and additionally performing a separate malicious activity
- Continuing to perform the function of the original program but modifying the function to perform malicious activity (e.g., a Trojan horse version of a login program that collects passwords) or to disguise other malicious activity (e.g., a Trojan horse version of a process listing program that does not display certain processes that are malicious)
- Performing a malicious function that completely replaces the function of the originalprogram

## ➢ Viruses

A computer virus is a piece of software that can "infect" other programs by modifying them; the modification includes injecting the original program with a routine to make copies of the virus program, which can then go on to infect other programs. A computer virus carries in its instructional code the recipe for making perfect copies of itself. The typical virus becomes embedded in a program on a computer. Then, whenever the infected computer comes into contact with an uninfected piece of software, a fresh copy of the virus passes into the new program. Thus, the infection can be spread from computer to computer by unsuspecting users who either swap disks or send programs to one another over a network. In a network environment, the ability to access applications and system services on other computers provides aperfect culture for the spread of a virus.

A virus can do anything that other programs do.The difference is that a virus attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs that is allowed by the privileges of the current user.

A computer virus has three parts :

- **Infection mechanism:** The means by which a virus spreads, enabling it toreplicate.

The mechanism is also referred to as the **infection vector**.

- **Trigger:** The event or condition that determines when the payload is activatedor delivered.
- **Payload:** What the virus does, besides spreading. The

## ➢ FIREWALLS

Firewalls were officially invented in the early 1990s, but the concept really reflects the reference monitor from two decades earlier.

**What is a Firewall?**

A firewall is a device that filters all traffic between a protected or "inside" network and a less trustworthy or "outside" network. Usually a firewall runs on a dedicated device; because it is a single point through which traffic is channeled, performance is important, which means non-firewall functions should not be done on the same machine. Because a firewall is executable code, an attacker could compromise that code and execute from the firewall's device. Thus, the fewer pieces of code on the device, the fewer tools the attacker would have by compromising the firewall. Firewall code usually runs on a proprietary or carefully minimized operating system. The purpose of a firewall is to keep "bad" things outside a protected environment. To accomplish that, firewalls implement a security policy that is specifically designed to address what bad things might happen. For example, the policy might be to prevent any access from outside (while still allowing traffic to pass *from* the inside *to* the outside). Alternatively, the policy might permit accesses only from certain places, from certain users, or for certain activities. Part of the challenge of protecting a network with a firewall is determining which security policy meets the needs of the installation.

**Design of Firewalls:**

A reference monitor must beAlways invoked Tamperproof

Small and simple enough for rigorous analysis

A firewall is a special form of reference monitor. By carefully positioning a firewall within a network, we

can ensure that all network accesses that we want to control must pass through it. This restriction meets the "always invoked" condition. A firewall is typically well isolated, making it highly immune to modification. Usually a firewall is implemented on a separate computer, with direct connections only to the outside and inside networks. This isolation is expected to meet the "tamperproof" requirement. And firewall designers strongly recommend keeping the functionality of the firewall simple.

**Types of Firewalls:**

Firewalls have a wide range of capabilities. Types of firewalls include

- □ Packet filtering gateways or screening routers
- □ Stateful inspection firewalls
- □ Application proxy
- □ Guards
- □ Personal firewalls

**Packet Filtering Gateway:**

A packet filtering gateway or screening router is the simplest, and in some situations, the most effective type of firewall. A packet filtering gateway controls access to packets on the basis of packet address (source or destination) or specific transport protocol type (such as HTTP web traffic). As described earlier in this chapter, putting ACLs on routers may severely impede their performance. But a separate firewall behind (on the local side) of the router can screen traffic before it gets to the protected network. Figure 7-34 shows a packet filter that blocks access from (or to) addresses in one network; the filter allows HTTP traffic but blocks traffic using the Telnet protocol.

**Stateful Inspection Firewall:**

Filtering firewalls work on packets one at a time, accepting or rejecting each packet and moving on to the next. They have no concept of "state" or "context" from one packet to the next. A stateful inspection firewall maintains state information from one packet to another in the input stream.

One classic approach used by attackers is to break an attack into multiple packets by forcing some packets to have very short lengths so that a firewall cannot detect the signature of an attack split across two or more packets. (Remember that with the TCP protocols, packets can arrive in any order, and the protocol suite is responsible for reassembling the packet stream in proper order before passing it along to the application.) A stateful inspection firewall would track the sequence of packets and conditions from one packet to another to thwart such an attack.

**Application Proxy**

Packet filters look only at the headers of packets, not at the data inside the packets. Therefore, a packet filter would pass anything to port 25, assuming its screening rules allow inbound connections to that port. But applications are complex and sometimes contain errors. Worse, applications (such as the e-mail delivery agent) often act on behalf of all users, so they require privileges of all users (for example, to store incoming mail messages so that inside users can read them). A flawed application, running with all users' privileges, can cause much damage. An application proxy gateway, also called a bastion host, is a firewall that simulates the (proper) effects of an application so that the application receives only requests to act properly. A proxy gateway is a two-headed device: It looks to the inside as if it is the outside (destination) connection, while to the outside it responds just as the insider would.

An application proxy runs pseudo-applications. For instance, when electronic mail is transferred to a location, a sending process at one site and a receiving process at the destination communicate by a protocol that establishes the legitimacy of a mail transfer and then actually transfers the mail message. The protocol between sender and destination is carefully defined. A proxy gateway essentially intrudes in the middle of this protocol exchange, seeming like a destination in communication with the sender that is outside the firewall, and seeming like the sender in communication with the real destination on the inside. The proxy in the middle has the opportunity to screen the mail transfer, ensuring that only acceptable

e-mail protocol commandsare sent to the destination.

**Guard:**

A guard is a sophisticated firewall. Like a proxy firewall, it receives protocol data units, interprets them, and passes through the same or different protocol data units that achieve either the same result or a modified result. The guard decides what services to perform on the user's behalf in accordance with its available knowledge, such as whatever it can reliably know of the (outside) user's identity, previous interactions, and so forth. The degree of control a guard can provide is limited only by what is computable. But guards and proxy firewalls are similar enough that the distinction between them is sometimes fuzzy. That is, we can add functionality to a proxy firewall until it starts to look a lot like a guard.

**Personal Firewalls:**

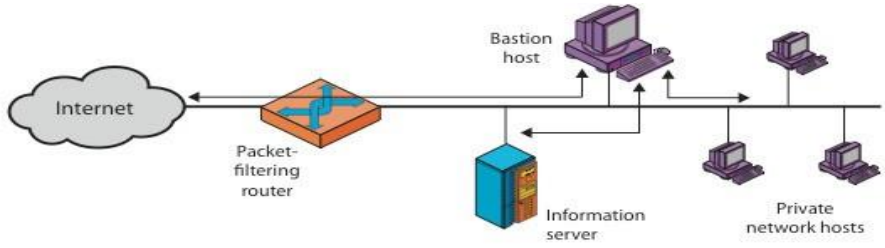
A personal firewall is an application program that runs on a workstation to block unwanted traffic, usually from the network. A personal firewall can complement the work of a conventional firewall by screening the kind of data a single host will accept, or it can compensate for the lack of a regular firewall, as in a private DSL or cable modem connection.

The personal firewall is configured to enforce some policy. For example, the user may decide that certain sites, such as computers on the company network, are highly trustworthy, but most other sites are not. The user defines a policy permitting download of code, unrestricted data sharing, and management access from the corporate segment, but not from other sites. Personal firewalls can also generate logs of accesses, which can be useful to examine in case something harmful does slip through the firewall.

A personal firewall runs on the very computer it is trying to protect. Thus, a clever attacker is likely to attempt an undetected attack that would disable or reconfigure the firewall for the future. Still, especially for cable modem, DSL, and other "always on" connections, the static workstation is a visible and vulnerable target for an ever- present attack community. A personal firewall can provide reasonable protection to clients that are not behind a network firewall.

**Firewall Configurations**

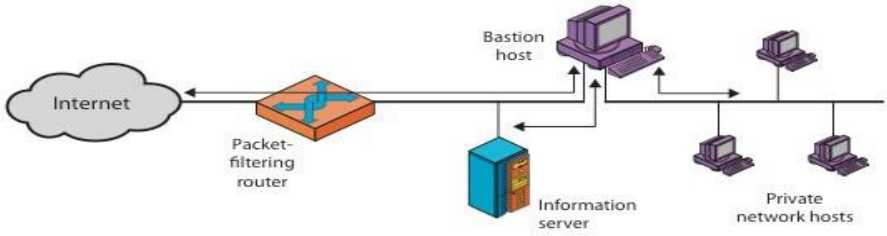
- In addition to the use of a simple configuration consisting of a single system, more complex configurations are possible and indeed more common.
- Figure illustrates three common firewall configurations.
- Figure (a) shows the "screened host firewall, single-homed bastion configuration", where the firewall consists of two systems:
- a packet-filtering router - allows Internet packets to/from bastion only
- a bastion host - performs authentication and proxy functions
- This configuration has greater security, as it implements both packet-level & application-level filtering, forces an intruder to generally penetrate two separate systems to compromise internal security, & also affords flexibility in providing direct Internet access to specific internal servers (eg web) if desired.




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

Figure (b) illustrates the "screened host firewall, dual-homed bastion configuration" which physically

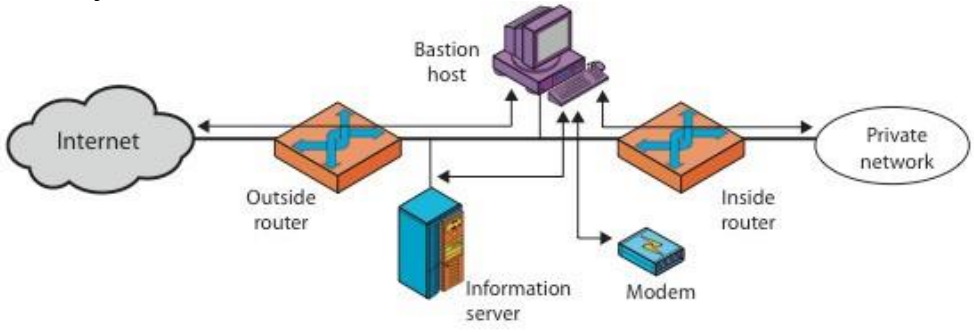
separates the external and internal networks, ensuring two systems must be compromised to breach security. The advantages of dual layers of security are also present here. Again, an information server or other hosts can be allowed direct communication with the router if this is in accord with the security policy, but are now separated from the internal network.
re(c) shows the "screened subnet firewall configuration", being the most secure shown. It has two packet-



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

filtering routers, one between the bastion host and the Internet and the other between the bastion host and the internal network, creating an isolated sub network. This may consist of simply the bastion host but may also include one or more information servers and modems for dial-in capability. Typically, both the Internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked.
This configuration offers several advantages:

- There are now three levels of defense to thwart intruders
- The outside router advertises only the existence of the screened subnet to the Internet; therefore the internal network is invisible to the Internet
- Similarly, the inside router advertises only the existence of the screened subnet to the internal network; hence systems on the inside network cannot construct direct routes to the Internet



(c) Screened-subnet firewall system

## ➢ WORMS

A computer worm is a type of malware that spreads copies of itself from computer to computer. A worm can replicate itself without any human interaction, and it does not need to attach itself to a software program in order to cause damage.
How do computer worms work?

Worms can be transmitted via software vulnerabilities. Or computer worms could arrive as attachments in spam emails or instant messages (IMs). Once opened, these files could provide a link to a malicious website or automatically download the computer worm. Once it's installed, the worm silently goes to work and infects the machine without the user's knowledge.

Worms can modify and delete files, and they can even inject additional malicious software onto a computer. Sometimes a computer worm's purpose is only to make copies of itself over and over —

depleting system resources, such as hard drive space or bandwidth, by overloading a shared network. In addition to wreaking havoc on a computer's resources, worms can also steal data, install a backdoor, and allow a hacker to gain control over a computer and its system settings.

How to tell if your computer has a worm

If you suspect your devices are infected with a computer worm, run a virus scan immediately. Even if the scan comes up negative, continue to be proactive by following these steps.

1. **Keep an eye on your hard drive space**. When worms repeatedly replicate themselves, they start to use up the free space on your computer.
2. **Monitor speed and performance**. Has your computer seemed a little sluggish lately? Are some of your programs crashing or not running properly? That could be a red flag that a worm is eating up your processing power.
3. **Be on the lookout for missing or new files**. One function of a computer worm is to delete and replace files on a computer.

How to help protect against computer worms

Computer worms are just one example of malicious software. To help protect your computer from worms and other online threats, take these steps.

1. Since software vulnerabilities are major infection vectors for computer worms, be sure your computer's operating system and applications are up to date with the latest versions. Install these updates as soon as they're available because updates often include patches for security flaws.
2. Phishing is another popular way for hackers to spread worms (and other types of malware). Always be extra cautious when opening unsolicited emails, especially those from unknown senders that contain attachments or dubious links.
3. Be sure to invest in a strong internet security software solution that can help block these threats. A good product should have anti-phishing technology as well as defenses against viruses, spyware, ransomware, and other online threats.

**Multipurpose Internet Mail Extensions (MIME)**

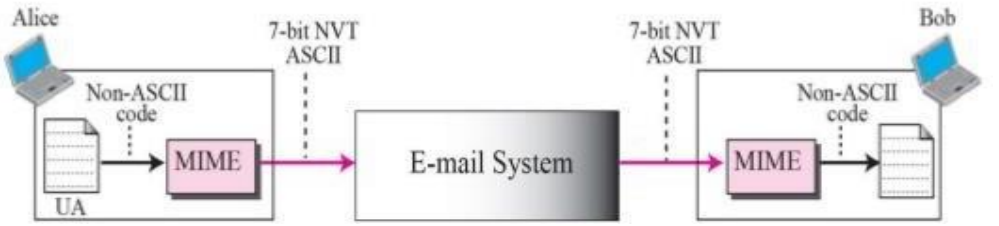
- **Multipurpose Internet Mail Extension (MIME)** is a standard which was proposed by BellCommunications in 1991 in order to expand limited capabilities of email.
- MIME is a kind of *add on or a supplementary protocol* which allows non-ASCII data to be sentthrough SMTP. It allows the users to exchange different kinds of data files on the Internet: audio, video, images, application programs as well.

- **Why do we need MIME?**

Limitations of Simple Mail Transfer Protocol (SMTP):

1. SMTP has a very simple structure
2. It's simplicity however comes with a price as it only send messages in NVT 7-bit ASCII format.
3. It cannot be used for languages that do not support 7-bit ASCII format such as-French, German, Russian, Chinese and Japanese, etc. so it cannot be transmitted using SMTP. So, inorder to make SMTP more broad we use MIME.
4. It cannot be used to send binary files or video or audio data.

•MIME was designed mainly for SMTP, but the content types defined by MIME standards are also of importance in communication protocols outside of email, such as Hyper Text Transfer Protocol (HTTP) for the World Wide Web.




MIME *transforms non-ASCII data* at sender side to NVT 7-bit data and delivers it to the client SMTP. The message at receiver side is transferred back to the original data. As well as we can send video and audio data using MIME as it transfers them also in 7-bit ASCII data.

**S/MIME (Secure/Multipurpose Internet Mail Extensions)**

- S/MIME is an extension of the widely implemented Multipurpose Internet Mail Extensions(MIME) encoding standard
- S/MIME uses the RSA public key cryptography algorithm along with the Data EncryptionStandard (DES) or Rivest-Shamir-Adleman (RSA) encryption algorithm.
- S/MIME provides the following functions:
- ● **Enveloped data:** This consists of encrypted content of any type and encrypted-contentencryption keys for one or more recipients.
- **Signed data:** A digital signature is formed by taking the message digest of the content to besigned and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.
- **Clear-signed data:** As with signed data, a digital signature of the content is formed. However, in this case, only the digital signature is encoded using base64. As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.

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

■ **Must:** The definition is an absolute requirement of the specification. An implementationmust include this feature or function to be in conformance with the specification.

■ ● **Should:** There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function.

## ➢ MOBILE DEVICE SECURITY

**Mobile Device Security** refers to the measures designed to protect sensitive information stored on and transmitted by laptops, smartphones, tablets, wearables, and other portable devices. At the root of mobile device security is the goal of keeping unauthorized users from accessing the enterprise network. It is one aspect of a complete enterprise security plan.

- Lackof Physical Security Controls

Mobile device is required to remain on premises, the user may move the device within the organization between secureand non secured locations. theft and tampering are realistic threats.

**The threat is two fold:**

1. A malicious party may attempt to recover sensitive data from the deviceitself
2. may use the device to gain access to the organization's resources.

- Use of Untrusted Mobile Devices

In addition to company-issued and company-controlled mobile devices, virtually all employees will have personal smartphones and/or tablets. The organization must assumethat these devices are not trustworthy.

- Use of Untrusted Networks

If a mobile device is used on premises, it can connect to organization resources over the organization's own in-housewireless networks.

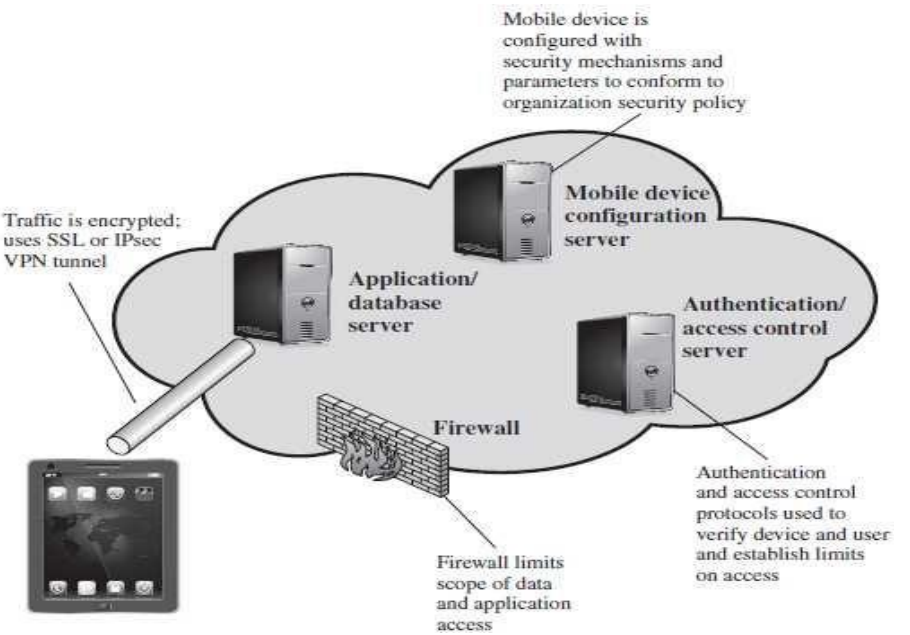
Thus, traffic that includes an off-premises segment is potentially susceptible to eavesdropping or man-in-the-middle types of attacks.

- Use of Applications Created by Unknown PartiesBy design, it is easy to find and install third-party

applications on mobile devices. This poses the obvious riskof installing malicious software.

- Interaction with Other Systems

Unless an organization has control of all the devices involvedin synchronization, there is considerable risk of the organization's data being stored in an unsecured location, plus the risk of the introduction of malware.




- **Use of Untrusted Content**

Mobile devices may access and use content that other computing devices do not encounter..

- Use of Location Services

The GPS service, it creates security risks. An attacker can use the location information to determine where the deviceand user are located, which may be of use to the attacker.

## ➢ Risk Model

cyber security risk modeling is the task of creating a variety of risk scenarios, assessing the severity of each, and quantifying the potential outcome if any scenario is realized – in a language that makes sense to your business.

Cyber risk modeling should not be confused with threat modeling. Threat model frameworks help identify cyber threats and vulnerabilities and inform and prioritize mitigation efforts. On the other hand, cyber risk modeling is an efficient and repeatable means of quantifying the likelihood of a cyber-attack. With this insight, your business can make robust decisions about where to focus investment for the greatest ROI.

**An example of cyber security risk modeling**

One of the most impactful examples of cyber security risk modeling is the quantification of cyber risk in financial terms as opposed to business terms. By establishing a universal understanding of cyber risk across your organization you can develop a more mature cybersecurity program and lead meaningful conversations on the business impact of different cyber scenarios and cybersecurity investments.

This analysis is not too different from the process of quantifying risk in a financial portfolio. For example, traders and portfolio managers use risk models to analyze and anticipate the impact of future events on performance so they can make preemptive decisions about where to invest funds.

A data-driven approach to understand risk exposure

Of course, any model is only as good as the data inputs and assumptions that go into it. The data must be current and accurately reflect the entire risk landscape. It's an overwhelming task for any security team. Digital ecosystems are expanding into the cloud and across business units and subsidiaries. It would take an army of resources to identify each digital asset, assess risk exposure, and calculate what a breach would mean financially.

The combined set of metrics delivers actionable analysis of cyber risk exposure across your business units, subsidiaries, and even M&A targets. And because no two risk scenarios are the same, you can simulate hundreds of thousands of events – ransomware, supply chain attacks, and more – and view the financial impact of each. You can also use these insights to diagnose the underlying vulnerabilities that impact financial exposure and inform what actions will deliver the greatest cyber risk reduction. Because risk is constantly evolving, the financial cyber risk quantification analysis is available on-demand and is easily repeatable so that you can measure risk exposure over time.

## ➢ Counter measure

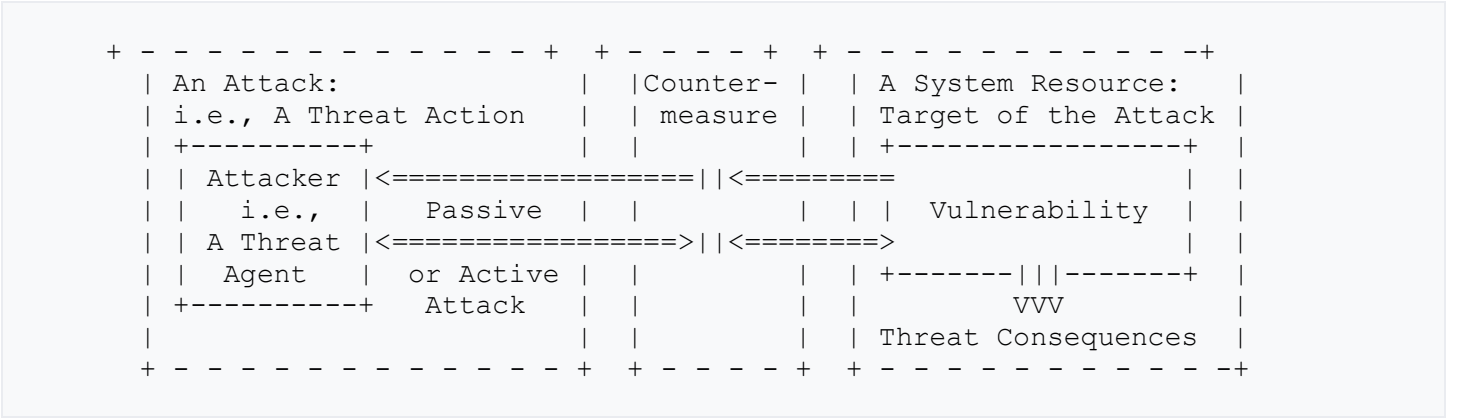
In computer security a **countermeasure** is an action, device, procedure, or technique that reduces a threat, a vulnerability, or an attack by eliminating or preventing it, by minimizing the harm it can cause, or by discovering and reporting it so that corrective action can be taken.

According to the Glossary meaning of countermeasure is:

The deployment of a set of security services to protect against a security threat.

A synonym is security control. In telecommunications, communication countermeasures are defined as security services as part of OSI Reference model

The following picture explain the relationships between these concepts and terms:



A resource (either physical or logical) can have one or more vulnerabilities that can be exploited by a threat agent in a threat action. The result can potentially compromises the confidentiality, integrity or availability properties of resources (potentially different that the vulnerable one) of the organization and others involved parties (customers, suppliers).
The so-called CIA triad is the basis of information security.

The attack can be active when it attempts to alter system resources or affect their operation: so it compromises integrity or availability. A "passive attack" attempts to learn or make use of information from the system but does not affect system resources, compromising confidentiality.

A threat is a potential for violation of security, which exists when there is a circumstance, capability, action, or event that could breach security and cause harm. That is, a threat is a possible danger enabling the exploitation of a vulnerability. A threat can be either "intentional" (i.e., intelligent; e.g., an individual cracker or a criminal organization) or "accidental" (e.g., the possibility of a computer malfunctioning, or the possibility of an "act of God" such as an earthquake, a fire, or a tornado).[1]

A set of policies concerned with information security management, the information security management systems (ISMS), has been developed to manage, according to risk management principles, the countermeasures in order to accomplish to a security strategy set up following rules and regulations applicable in a country

**Electronic Destruction Devices**

Devices such as a USB Killer may be used to damage or render completely unusable anything with a connection to the motherboard of a computer, such as a USB port, video port, Ethernet port, or serial port. Without proper protection, these devices may result in the destruction of ports, adapter cards, storage devices, RAM, motherboards, CPUs, or anything physically connected to the device attacked, such as monitors, flash drives, or wired switches. These types of devices can even be used to damage smartphones and cars, as well.

This threat can be mitigated by not installing or restricting physical access to easily accessible ports in situations where they are not necessary. A port-closing lock which permanently disables access to a port short of the actual port being disassembled. When it is necessary for a port to be accessible, an opto coupler can allow for a port to send and receive data to a computer or device without a direct electrical connection, preventing the computer or device from receiving any dangerous voltage from an external device.

**Hard Drives and Storage**

In an unsecured scenario, a malicious actor may steal or destroy storage devices such as hard drives or SSDs, resulting in the destruction or theft of valuable data.

If the data of a storage device is no longer necessary, data theft is best prevented against by physically destroying or shredding the storage device.

If the data of a storage device is in use and must be secured, one can use encryption to encrypt the contents of a storage device, or even encrypt the whole storage device save for the master boot record. The device can then be unlocked with a password, biometric authentication, a physical dongle, a network interchange, a one-time password, or any combination thereof. If this device is a boot drive, however, it must be unencrypted in a pre-boot environment so the operating system can be accessed. Striping, or breaking data into chunks stored upon multiple drives which must be assemble in order to access the data, is a possible solution to physical drive theft, provided that the drives are stored in multiple, individually secured locations, and are enough in number that no one drive can be used to piece together meaningful information.

Not to be neglected is the process of adding physical barriers to the storage devices themselves. Locked cases or physically hidden drives, with a limited number of personnel with knowledge and access to the keys or locations, may prove to be a good first line against physical theft.

What is Cloud Cryptography?

According to privacy experts, cryptography is the cornerstone of security. Cloud cryptography is the encryption of data stored in the cloud which adds a strong layer of protection and avoids a data breach. This practice safeguards data without delaying the data delivery. Cryptography expert Ralph Spencer Power said "information in motion and information at rest can be better protected by cryptography. Virtual data needs to be stored cryptographically by maintaining the control of the cryptographic key." Now, let us learn how to implement cryptography in the cloud and protect our cloud data.

How to Implement cryptography in Cloud Computing?

It's not possible to control cloud data physically. Cloud encryption is a way of protecting data and communication with the help of codes. Cloud data encryption can guard sensitive data and verify asset transfer without delaying the information transmission. Several tech giants like Google and Amazon define cryptographic protocols for their cloud computing to balance efficiency and security.

There are different kinds of cryptographic keys used by companies for **cloud security**. Cloud data encryption depends on three algorithms:

1. Symmetric-key
2. Asymmetric key
3. Hashing

**Symmetric Algorithm**

It utilizes one key for both encryption and information decoding. It doesn't need a lot of computational power and works extremely high in encryption. Symmetrical algorithms consist of two-way keys to guarantee verification and approval. Except if the client has the key, the encoded information is put away in the Cloud, and can't be decoded.

Some popular symmetric algorithms used in cloud computing algorithms are – Data encryption standard (DES), Blowfish, Advanced encryption standard, Triple DES, etc.

- **Advanced Encryption Standard (AES)** – It is used to encrypt digital data such as telecommunications, financial, and government. In AES, the same key is used for both encryption and decryption. It is a block of ciphertext that repeats itself after every defined step multiple times. It has a 128-bit block size, with key sizes of 128, 192, and 256 bits. It's efficient in both software and hardware.
- **Data Encryption Standard (DES)** – It adopts a 64-bit secret key, out of which 56 bits are created randomly and the remaining 8 bits are used for error detection. DES is implemented in hardware and is basically used for single-user encryption, for eg – files stored on a hard disk in encrypted form.

**Asymmetric Algorithm**

It utilizes different keys for encryption and decoding. Here, every beneficiary requires a decoding key. This key is referred to as the recipient's private key. Here, the encryption key belongs to a particular individual or entity. This sort of algorithm is considered the most secure as it requires both keys to get to a piece of explicit data.

- **Rivest Shamir Adleman Algorithm (RSA)** – It is one of the de-facto encryption standards and is used for a variety of platforms. It used different keys for encryption and decryption. The public key is known to everyone which can be decrypted using the private key only by the authorized person.
- **Elliptic Curve Cryptography (ECC)** – ECC is modern public-key cryptography that depends on number theory and mathematical elliptic curves to generate a short key. ECC is preferred by the security experts because of the small key size of the ECC.

**Hashing**

It is one of the major parts of block chain security. In the block chain, data is put away in blocks and interconnected with cryptographic standards like a string or chain. When an information block is added to

the chain, a particular code or hash is assigned to the particular block. Hashing is basically utilized for ordering and recovering things in a data set. It likewise utilizes two distinctive keys for encrypting and decoding a message. It likewise gives quicker information retrieval.

Advantages of cloud cryptography

- cryptography in the cloud is probably the most secure strategy to store and move the information as it complies with the limitations forced by associations like FIPS, FISMA, HIPAA, or PCI/DSS.
- The information stays private to the clients. cryptography in the cloud lessens the cybercrime cases.
- Companies get warnings promptly if an unauthorized individual attempts to access the data. The clients who have cryptographic keys are only allowed data admittance.
- The encryption keeps the information from being vulnerable when the information is being transferred from one PC then onto the next,
- Cloud encryption prepares organizations to stay proactive with all due respect against cyberattacks
- Receivers of the data can recognize if the information got is adulterated, allowing a prompt reaction and answer for the assault.

Disadvantages of cloud cryptography

- Cloud cryptography just grants restricted security to the information which is in transit.
- Cloud encryption needs exceptionally advanced systems to maintain encrypted information.
- The frameworks should be adequately versatile to update which adds to the involved costs.

Companies and organizations need to take a data-centric approach to protecting their sensitive information in order to guard against advanced threats in the complex and evolving environments of virtualization, cloud services, and mobility. Companies should implement data security solutions that provide consistent protection of sensitive data, including cloud data protection through encryption and cryptographic key management. A comprehensive platform for cloud security and encryption also should deliver robust access controls and key management capabilities that enable organizations to practically, cost effectively, and comprehensively leverage encryption to address security objectives.

6 Significant Cloud Security Threats

Organizations and businesses have had to turn to third-party cloud and managed security services to look for ways to bolster cybersecurity and shift from legacy to modern data platforms.

However, the sudden transition to the cloud has brought new security risks. This means that if your business or organization chooses to adopt cloud technologies and migrate your data over, you could be making a major mistake without being fully informed of the risks involved.

In this article, we will outline the six most significant cybersecurity threats for cloud networks that businesses face when migrating data or applications to the cloud. Take note that these cloud security threats are always evolving and the ones listed here are by no means exhaustive.

## ➢ Cloud Security Threats

**Data Breaches**

Data breaches occur when unauthorized individuals access cloud systems and interfere with the data stored in them. Whether attackers view, copy or transmit data, an organization's safety is not guaranteed once such individuals gain access.

Significant data breaches that have been costly to businesses include the mid-2018 Tesla cloud crypto-jacking that exposed sensitive telemetry data. This occurred due to the company's failure to encrypt one of its cloud accounts.

The primary cause of data breaches is human error. Lack of knowledge or not educating your staff on how to keep data safe and secure can easily expose your business to a hacker. This is why providing sufficient cybersecurity education on data protection to your employees is crucial, as nearly 90% of professionals agree that improved data protection skills can significantly reduce risks and data breaches happening within their respective organizations.

**Insider Threats**

Sometimes, the biggest threats to an organization's cybersecurity are internal. Insider threats are usually seen as more hazardous than outsider threats as they can take several months or years to identify.
The masterminds are usually individuals with legitimate access to an organization's cloud systems. Whether they happen intentionally or maliciously, insider threats will cause a lot of harm to your cloud system. Therefore, it is essential to detect, investigate and respond to them as fast as possible.

The reason why these attacks can go undetected for long periods is that businesses lack the proper systems to identify these attacks and are unprepared to identify and resolve them. In addition, companies have little to no control over underlying cloud infrastructure. Traditional security solutions may not be effective as long as significant power remains with the vendors.

Monitoring user analytics and gaining visibility into behavioral anomalies can be a way to signal an active insider threat as well as putting employees and processes to the test with adversary simulation and control tuning.

**Denial-of-Service Attacks**

Due to the rise of cyberattacks brought on by the global pandemic, an increasing number of companies are shifting their data control to the cloud. However, this leaves most applications and essential internal functions that are cloud-based exposed to denial-of-service attacks.
In a denial-of-service attack, a hacker floods a system with more web traffic than it can handle at its peak. This results in operations stalling entirely, with internal users and customers unable to access the system, making it unable to operate the business.

Subsequently, companies need to find ways to stop denial-of-service attacks before they occur and cause serious setbacks. One strategy is to rely on dynamic application security tools, which will scan your web applications for threats while they are running and can identify denial-of-service attacks in their early stages or before they happen.

**Insecure Interfaces and APIs**

Software user interfaces and APIs are usually responsible for the provision, monitoring and management of cloud services. Cloud service providers are working tirelessly to advance APIs and interfaces, but this growth has also increased security risks associated with them.

Cloud service providers use a specific framework to provide APIs to programmers, which leaves their systems more vulnerable to attackers. As such, organizations risk improper authorizations, previously used passwords and anonymous access. The best way to solve this is knowing how to properly design your cloud security with a multi-layer approach, which is required to help curb unauthorized access and ensure that the software you create is secure.

**Hijacking of Accounts**

The growing reliance on cloud-based infrastructure has also contributed to a high number of account hijacking cases. Depending on the attacker's intent and how they will use the accessed information, cloud account hijacking can have devastating consequences for a business, such as information being falsified or leaked to other parties.
Account hijacking attacks can also damage a brand's reputation and the relationships they have with their customers. The integrity and good reputation a company has built for years can be destroyed with one cyberattack. Legal implications could also follow if customers decide to sue the company for exposing their confidential data.

Having rock-solid facilities that utilize electronic surveillance and multifactor access systems is important to minimize the risk of hijacking and disruptions to operations. Having a provider that also offers features such as secure data transfer, encrypted data storage and security logs will provide detection of brute-force attacks.

**Misconfigurations**

Misconfigurations is one of the leading threats businesses face in their cloud-based systems. Most business owners are inexperienced in matters surrounding cloud-based infrastructure, which exposes them to various data breaches that can impact their operations.

Misconfigurations often results from the need to make cloud data accessible and shareable. Limiting access only to eligible people and, depending on the cloud service provider, can impact a company's ability to control these systems dramatically. Basic cloud storage services often come with critical security measures such as client-side encryption, intrusion detection systems and internal firewalls. Being familiar with vendor-provided security settings is critical

## ➢ Security at service layers

When some enterprises migrate to the cloud, they wrongly assume that workload security is now in the hands of their cloud provider.

In reality, most cloud vendors enforce what's called a shared responsibility model. This model varies depending on the cloud computing service category -- SaaS, PaaS or IaaS -- but, in all cases, security responsibilities are split to some degree between the cloud provider and its users.

When applications and servers are hosted in-house, IT operations admins' security responsibilities are clearly defined; teams can physically see, or at least have direct control over, the IT resources that run in their data center. With cloud computing, however -- where users essentially "rent" compute resources from a provider admins must drastically change how they manage workloads. And, in some cases, this creates gaps in security coverage.

While SaaS and PaaS each present unique cloud security considerations, admins can also apply some key best practices from their days of securing on-premises resources.

**SaaS security emphasizes access control**

With SaaS, enterprises access an application that is fully hosted and managed by a cloud provider. It might appear as if IT teams are free from any security responsibilities, particularly when compared to how they maintain on-premises workloads. The problem, however, is that this is an apples-to-oranges comparison. IT teams still need to manage configurations and access controls for SaaS applications.

The SaaS provider does manage and secure the infrastructure, OS and application stack -- but IT teams still need to manage configurations and access controls for SaaS applications. Most SaaS offerings -- whether Microsoft Office 365 or a learning management system or HR tool -- come with admin accounts, from which IT staff can add or remove user access permissions for the application. Admins also can enable or disable certain application features to fit the enterprise's needs and compliance model. Limit access to these administrative accounts to only a select group of operations admins, and where possible, make the admin accounts separate from their daily user accounts to avoid accidental SaaS-wide changes.

A cloud migration is a complex process, during which IT teams might grant permissions on a temporary basis -- and they can forget to reset access after the migration is complete. Account auditing is an essential cloud security consideration for all applications hosted there.

**PaaS security brings greater responsibility**

Compared to SaaS, IT staff's responsibilities increase with PaaS deployments. PaaS grants admins more control over the application stack -- which shifts more security responsibilities from the cloud provider to the user.

PaaS security can be a challenge, organizationally, for operations staff, since the team that owns the application typically handles application security, rather than the security and infrastructure team. In other words, PaaS puts a lot of security responsibility onto people whose primary concern is application delivery.

Enterprises have more security responsibilities for PaaS than they do with SaaS.

This gap between the application owner and the security teams has always existed, but it becomes more evident with cloud adoption. Ops teams may find they have another hat to wear; while they won't be responsible for securing the cloud application itself, they will have to enforce -- and verify -- that other parties follow security best practices.

A change to toolsets -- and processes

Another cloud security consideration is that ops and the security teams can't use traditional tools and processes for cloud security testing and verification. For example, in some cases, enterprise IT teams must notify their cloud provider when they plan to run security scans or penetration tests on that provider's

resources. Even if a cloud provider does not require these notifications, it generally defines certain practices that users must follow to perform these tests. In addition, a cloud provider's internal security teams have the right to respond to tests performed on the platform.

Cloud providers' native tools, such as AWS Security Hub and Azure Security Center, along with several third-party products, such as HyTrust and CipherCloud, can validate cloud deployment security. Cloud providers don't want to see their users experience a security issue -- because it's simply bad for business -- so take advantage of the tools they offer.

Overall, though, cloud security considerations are less about technical shortcomings, and more about processes and verification. IT ops teams need to emphasize policy and procedure -- this will go much further to secure a cloud deployment than any one tool can.

## ➢ Introduction to Block chain

Think of a block chain as a novel, digital form of record-keeping.

Blockchain is the underlying technology that many cryptocurrencies — like Bitcoin and Ethereum — operate on, but its unique way of securely recording and transferring information has broader applications outside of cryptocurrency.

A block chain is a type of distributed ledger. Distributed ledger technology (DLT) allows record keeping across multiple computers, known as "nodes." Any user of the block chain can be a node, but it takes a lot of computer power to operate. Nodes verify, approve, and store data within the ledger. This is different from traditional record-keeping methods which store data in a central place, such as a computer server.

A block chain organizes information added to the ledger into blocks, or groups of data. Each block can only hold a certain amount of information, so new blocks are continually added to the ledger, forming a chain.

Each block has its own unique identifier, a cryptographic "hash." The hash not only protects the information within the block from anyone without the required code, but also protects the block's place along the chain by identifying the block that came before it.

Once information is added to the block chain and encrypted with a hash, it's permanent and unchangeable. Each node has its own record of the full timeline of data along the block chain, going back to its start. If someone tampered with or hacked into one computer and manipulated the data for their own gain, it wouldn't alter the information stored by other nodes. The altered record can be easily distinguished and corrected, since it doesn't match the majority.

"The way that the system works, it's almost impossible for someone to replicate the computing power that happens on the back end to reverse engineer it, and somehow figure out what all those hashes are," Agarwal says.

**How it Works**

Here's an example of how block chain is used to verify and record Bitcoin transactions.

- A consumer buys Bitcoin.
- The transaction data is sent across Bitcoin's decentralized network of nodes.
- Nodes validate the transaction.
- After approval, the transaction is grouped with other transactions to form a block, which is added to an ever-growing chain of transactions.
- The completed block is encrypted, and the transaction record is permanent; it cannot be removed or altered on the block chain.

Bitcoin's block chain is public, which means anyone who owns Bitcoin can view the transaction record. While it can be difficult to trace the identity behind an account, the record shows which accounts are transacting on the block chain. Public block chains also allow any user with the required computer power to participate in approving and recording transactions onto the block chain as a node.

But not all block chains are public. Block chains can be designed as private ledgers, so an owner is able to limit who can make changes or additions to the block chain. While the pool of participants may be smaller

on a private block chain, it's still decentralized among those who participate. Private block chains maintain the security of any data stored within the database using the same encryption methods.

The idea of a secure, decentralized permanent record of information has drawn interest across a number of industries, and potentially holds solutions for many security concerns, record-keeping processes, and data ownership issues we face today.

## Innovative applications of blockchain

You don't need to look far to come up with a list of innovative ways in which blockchain technology is being applied. A wide range of fields, including healthcare, real estate, government, and music are finding a use for blockchain's powerful, secure way of storing, verifying, and encrypting data.8 Here are seven more applications of blockchain technology, some of which are underscored by cryptocurrencies:

**Finance**

One of the main services of the financial sector is to store money and transfer it from one entity to another. This requires a trustworthy intermediary, in the form of a bank. Blockchain is now effectively eliminating the need for such intermediaries by decentralizing transactions. By moving the means of transaction out of siloed, closed networks, blockchain is helping to solve some of the challenges around the interoperability of disparate financial systems around the world.9.The ability to track all transactions increases the transparency and security of blockchain-based payments too. This is beneficial both to the parties of a transaction and to relevant regulators.10

**Smart contracts**

Smart contracts act as self-executing programs, triggered automatically when predetermined conditions are met, that facilitate the terms of agreement between the seller and buyer directly. As they are executed on a blockchain network, the transactions are trackable, transparent, and irreversible.This type of automation can significantly boost productivity while slashing costs in business. Put simply, it helps you exchange property, shares, legal documents, or anything of value in a manner that's transparent and free of conflict, while also avoiding the expense of a middleman.[12]

**Cybersecurity**

Data stored on blockchain is rendered tamper-proof because the network of nodes (the disparate computers on which the shared database is stored, and which validate transactions) can cross-reference to locate the source of a disputed change, so the technology has a number of potential cybersecurity applications. Storing information across a network of devices reduces the risk of a hacker exploiting a single point of vulnerability. Similarly, decentralizing control of edge devices (which provide an entry point into enterprise or service provider core networks) and Internet of Things devices can render these devices more secure against attacks.

**Health records**

A decentralized, secure, trustworthy blockchain system has clear applications for the storage of healthcare records. Personal health records (PHR) collect data from sources including medical centers, devices, clinics, and pharmacies, and are primarily managed by patients. Electronic health records (EHR) are digital records of patients' medical history, and are managed by doctors.

As patients manage PHRs, the validity of that information is sometimes doubted. Storing them on block chains would ensure they are traceable, transparent, unchangeable, auditable, and secure.

EHRs, on the other hand, are typically stored in centralized legacy systems, which may not be interoperable between different healthcare facilities. Blockchain could offer a solution, with EHRs stored securely on a decentralized system that can be accessed – by both patients and healthcare workers – across systems and organizations.

**Non-fungible tokens**

NFTs, as they're more commonly known, are tokens on block chains, but they differ from cryptocurrencies in that they are unique digital assets.16 Technically, NFTs can represent ownership of anything, but they are mostly used to buy and sell digital art. In many cases, this digital art already exists

and is freely available on the internet for anyone to view, buy, or download. What an NFT confers is ownership of that art. Think of it as the difference between owning an original painting and a print of it.17

Voting

Recording and storing high-value, high-volume pieces of data is inherent to the voting process. This makes blockchain an ideal technology for updating the voting system. Because all nodes on a blockchain must verify any information entered onto it, people could potentially cast their votes online without fear of fraud. It also creates greater certainty for electoral officials, who can tally votes certain in the knowledge that each is attributable to only one individual.

- A **cryptocurrency**, **crypto-currency**, or **crypto** is a digital currency designed to work as a medium of exchange through a computer network that is not reliant on any central authority, such as a government or bank, to uphold or maintain it.

Individual coin ownership records are stored in a digital ledger, which is a computerized database using strong cryptography to secure transaction records, to control the creation of additional coins, and to verify the transfer of coin ownership. Despite their name, cryptocurrencies are not necessarily considered to be currencies in the traditional sense and while varying categorical treatments have been applied to them, including classification as commodities, securities, as well as currencies, cryptocurrencies are generally viewed as a distinct asset class in practice. Some crypto schemes use validators to maintain the cryptocurrency. In a proof-of-stake model, owners put up their tokens as collateral. In return, they get authority over the token in proportion to the amount they stake. Generally, these token stakers get additional ownership in the token over time via network fees, newly minted tokens or other such reward mechanisms.

Cryptocurrency does not exist in physical form (like paper money) and is typically not issued by a central authority. Cryptocurrencies typically use decentralized control as opposed to a central bank digital currency (CBDC). When a cryptocurrency is minted or created prior to issuance or issued by a single issuer, it is generally considered centralized. When implemented with decentralized control, each cryptocurrency works through distributed ledger technology, typically a blockchain that serves as a public financial transaction database.

A cryptocurrency is a tradable digital asset or digital form of money, built on blockchain technology that only exists online. Cryptocurrencies use encryption to authenticate and protect transactions, hence their name. There are currently over a thousand different cryptocurrencies in the world.

Bitcoin, first released as open-source software in 2009, is the first decentralized cryptocurrency. Since the release of bitcoin, many other cryptocurrencies have been created.

## BitCoinSecurity and Working

Bitcoin, often described as a cryptocurrency, a virtual currency or a digital currency - is a type of money that is completely virtual.

It's like an online version of cash. You can use it to buy products and services, but not many shops accept Bitcoin yet and some countries have banned it altogether.

However, some companies are beginning to buy into its growing influence.

Bitcoin is a type of digital token that can be sent electronically through a decentralized digital payment network. Bitcoins can be sent from person to person, anywhere in the world; indeed, Bitcoin was initially intended to be used as a secure electronic cash and payment system.

Bitcoin is built on blockchain technology. A blockchain is a type of digital ledger that records information (such as transactions) in a way that makes it nearly impossible to edit or alter that information. This way of recording information is inherently secure, but Bitcoin takes it a step further by specifically employing *a* decentralized blockchain, which depends on a peer-to-peer network to verify transactions.

**How does Bitcoin work?**

Each Bitcoin is basically a computer file which is stored in a 'digital wallet' app on a smartphone or computer.
People can send Bitcoins (or part of one) to your digital wallet, and you can send Bitcoins to other people.
Every single transaction is recorded in a public list called the blockchain.
Bitcoin mining is the process by which Bitcoin transactions are validated. It's *also* the process by which new Bitcoins enter circulation. Allow us to explain.
We just mentioned that Bitcoin's consensus model requires a ton of computing power to function. This consensus model is called "proof-of-work," and it's integral to an understanding not only of how Bitcoin transactions are verified, but also of how new Bitcoins are created.
Bitcoin's "proof-of-work" model requires miners on the Bitcoin network to solve highly complex math problems to validate transactions. In return, these miners are rewarded with newly created Bitcoins. The fact that so many computers are spending so much power to validate transactions means that it's essentially impossible to get at least 51% of those computers to validate an inaccurate version of the ledger.

**What are Bitcoins used for?**

Though it was originally conceived of as a cash payment system, Bitcoin has grown into a number of different uses. Here are a few:

- You can use Bitcoin to buy things. From glamorous luxury cars to everyday insurance, you can use Bitcoin to buy all kinds of things. And with Bitcoin debit cards, which are loaded with cryptocurrency but are also capable of completing day-to-day transactions in fiat currency, you can "use" Bitcoin anywhere that accepts plastic.
- You can consider Bitcoin as a store of value. Though it's a far cry from typical investments, Bitcoin is also considered by many as an appealing store of value. Its volatile, whiplash pricing means that Bitcoin is a highly risky asset, but that hasn't stopped many speculators from piling in. The total number of Bitcoins is capped, which encourages some to see it as "digital gold."
- You can buy, sell, and trade Bitcoin. Due to their volatile and unpredictable pricing on the open market, Bitcoin and other cryptocurrencies have become popular with day traders and investors alike. Keep in mind, though, that any investment in cryptocurrency carries with it serious risks.

**Advantages of Bitcoin**

- Bitcoin is by its very nature secure, with little risk of false or double-spending transactions being verified by the network.
- Bitcoin is a highly transparent financial vehicle, with every transaction recorded in the blockchain for all to see.
- Bitcoin offers potential for major returns thanks to its high price volatility—though this also comes with significant risk

**Disadvantages of Bitcoin**

- Bitcoin's volatility can also be seen as one of its chief disadvantages, especially if you plan to use it as a store of value.
- Bitcoin isn't yet ready to replace cash for day-to-day needs.
- Bitcoin mining is an energy-intensive process that requires expensive equipment. This makes Bitcoin less appealing to environmentalists and those concerned about climate change.

## ➢ Ethereum

**Ethereum** isa decentralized, open-source blockchain with smart contract functionality. **Ether** (**ETH** or **Ξ**) is the native cryptocurrency of the platform. Among cryptocurrencies, Ether is second only to Bitcoin in market capitalization.

Ethereum allows anyone to deploy permanent and immutable decentralized applications onto it, with which users can interact. Decentralized finance (DeFi) applications provide a broad array of financial services without the need for typical financial intermediaries like brokerages, exchanges, or banks, such as allowing cryptocurrency users to borrow against their holdings or lend them out for interest. Ethereum also allows users to create and exchange NFTs, which are non-interchangeable tokens connected to digital

works of art or other real-world items and exchanged as a variety of digital property. Additionally, many other cryptocurrencies operate as ERC-20 tokens on top of the Ethereum blockchain and have utilized the platform for initial coin offerings.

Ethereum is permissionless non-hierarchical network of computers (nodes) that build and come to a consensus on an ever-growing series of "blocks", or batches of transactions, known as the blockchain. Each block contains an identifier of the chain that must precede it if the block is to be considered valid. Whenever a node adds a block to its chain, it executes the transactions therein order, thereby altering the ETH balances and other storage values of Ethereum accounts. These balances and values, collectively known as the "state", are maintained on the node separately from the block chain,

Each node communicates with a relatively small subset of the network—its "peers". Whenever a node wishes to include a new transaction in the blockchain, it sends the transaction to its peers, who then send it to their peers, and so on. In this way, it propagates throughout the network. Certain nodes, called miners, maintain a list of all of these new transactions and use them to create new blocks, which they then send to the rest of the network. Whenever a node receives a block, it checks the validity of the block and of all of the transactions therein and, if it finds the block to be valid, adds it to its blockchain and executes all of those transactions. Since block creation and broadcasting are permissionless, a node may receive multiple blocks competing to be the successor to a particular block. The node keeps track of all of the valid chains that result from this and regularly drops the shortest one: According to the Ethereum protocol, the longest of multiple competing chains is to be considered the canonical one.

**Ether**

Ether (ETH) is the cryptocurrency generated by the Ethereum protocol as a reward to miners in a proof-of-work system for adding blocks to the blockchain. It is the only currency accepted to pay for transaction fees, which also go to miners. The block-addition reward together with the transaction fees provides the incentive to miners to keep the blockchain growing (i.e. to keep processing new transactions). Therefore, ETH is fundamental to the operation of the network. Each Ethereum account has an ETH balance and may send ETH to any other account. The smallest subunit of ETH is known as a Wei, named after cryptocurrency.

**Accounts**

There are two types of accounts on Ethereum: user accounts (also known as externally-owned accounts) and contracts. Both types have an ETH balance, may send ETH to any account, may call any public function of a contract or create a new contract, and are identified on the blockchain and in the state by an account address.

User accounts are the only type of account that may create transactions. For a transaction to be valid, it must be signed using the sending account's private key, the 64-character hexadecimal string from which the account's address is derived. The algorithm used to produce the signature is ECDSA. Importantly, this algorithm allows one to derive the signer's address from the signature without knowing the private key.

Contracts are the only type of account that have associated code (a set of functions and variable declarations) and contract storage (the values of the variables at any given time). A contract function may take arguments and may have return values. Within the body of a function, in addition to control flow statements, a contract's code may include instructions to send ETH, read from and write to its storage, create temporary storage (memory) that vanishes at the end of the function, perform arithmetic and hashing operations, call its own functions, call public functions of other contracts, create new contracts, and query information about the current transaction or the blockchain.

## ➢ Ecosystem

### What is a blockchain ecosystem?

A blockchain ecosystem is a group of various technological elements capable of interacting to create a system that performs a specified function. This system encompasses multiple governing structures like individual participation, data ownership, funding, exit and entrance criteria, information shared with the system's participants.

The blockchain ecosystem can provide true decentralisation, immutability, transparency, accountability and flexibility to day-to-day operations.

These advantages can be a boon for technology startups and projects as it creates an interconnected network, even if they must consider what information they want to share within the network.

**The types of blockchain ecosystems**

A blockchain ecosystem could have participants in its network with different goals and business models. They can even have distinct contributions to the network. In fact, the various participants can even be competitors. In a blockchain ecosystem, every participant would be looking out for themselves and what business value they receive as a part of the ecosystem.

The type of blockchain ecosystem for a shared blockchain project will depend on the participants in the network. That is why different types of blockchain ecosystems accommodate the specific needs of the participants in the network. Here are some of the notable ones being used these days.

**Single party-led:** A Single party blockchain project is led by a single organisation where its stakeholders have a mutual benefit for participating in the network. An example of this ecosystem would be Bumble Bee Foods. They created an ecosystem of various stakeholders to improve the traceability of yellowfin tuna fish from the ocean to a customer's dinner table.

The stakeholder of Bumble bee Foods includes fishermen, packagers, transportation personnel, distributors, and retailers who record the fish's details on the blockchain. Customers can use a QR code to view this information. It can help improve the buyer's confidence in the fish's freshness.

**Joint venture ecosystem:** Joint ventures ecosystems usually have two or more organisations at the helm. These ecosystems are slowly becoming popular and are now overtaking formal joint ventures. These ecosystems are created to pool sources for a common goal.

The only question in the ecosystem is whether the organisations forming a strategic business association are a new legal entity or just entering a formal contractual arrangement.

An example of a joint venture ecosystem is BunkerTrace, an association between Forecast Technology Ltd. and BLOC (Blockchain Labs for Open Collaboration). BunkerTrace is a marine fuel tracking solution.

**Regulatory blockchain ecosystems:** This ecosystem comprises various government agencies that share a project and have to self-report for compliance. An example of a regulatory ecosystem would be the shared project between Marine Transport International and the Recycling Association. They use a blockchain-based tool to capture data concerning shipments of recyclable waste from Britain.

A crucial aspect of all these ecosystems is how they will be funded. Typically, ecosystems are excluded from the day-to-day business model funding. An organisation has to set aside a budget for creating a blockchain ecosystem as they have to consider various governance and operations costs for the ecosystem.

The rapid growth of the crypto ecosystem presents new opportunities. Technological innovation is ushering in a new era that makes payments and other financial services cheaper, faster, more accessible, and allows them to flow across borders swiftly. Crypto asset technologies have potential as a tool for faster and cheaper cross-border payments. Bank deposits can be transformed to stable coins that allow instant access to a vast array of financial products from digital platforms and allow instant currency conversion. Decentralized finance could become a platform for more innovative, inclusive, and transparent financial services. Despite potential gains, the rapid growth and increasing adoption12 of crypto assets also pose financial stability challenges. This chapter discusses the implications of the expansion of the crypto ecosystem and provides an assessment of their associated financial stability risks. For emerging market and developing economies, greater use of crypto assets presents some benefits, but

also macro-financial risks, especially with respect to asset and currency substitution—referred to in this chapter as cryptoization.

## ➢ Service Risk

**Mobile Risks and Attacks** Mobile applications are implemented in many of the same languages as their desktop and Web counterparts (e.g., Objective-C and Swift for iOS, Java for Android), and therefore are susceptible to many of the same vulnerabilities and attacks associated with those languages including infection and compromise by malicious software including spyware, Trojan horse programs, worms, and computer viruses. Phishing and other social engineering tactics prey on the weaknesses of the users. Other attacks target the mobile application itself, the server to which the mobile application speaks, unprotected internal APIs, alternate routes through and around security checks, and open server ports.

As with any technology channel, there are risks and threats associated with the mobile platform. It is possible for organizations to mitigate the potential risks and vulnerabilities during the development of mobile applications by establishing (or optimizing) formal security policies, procedures, and standards; education and training; and security engineering activities. These strategies are discussed in more detail in Mobile Risk and Threats – Part Two: Implementing Countermeasures.

**MOBILE-SPECIFIC CHALLENGES**

Due to the nature of how mobile devices function, they tend to have unique vulnerabilities when compared to desktops and servers, each with its own idiosyncrasies, built-in defenses, attack vectors, and threats. For example:

- **Physical Size** – The small size of mobile devices enables their distinctive portability – but makes them more susceptible to being lost, stolen, or temporarily misplaced. Once a mobile device is out of the authorized user's hands the device's stored user credentials, personal and corporate data, and gateway applications (such as VPN) for authorized access to corporate systems are at risk. In addition, the smaller screens and keyboards of mobile devices often force developers to make security trade-offs to accommodate a better mobile user experience.

- **Mobile Connections** – Mobile devices connect through a variety of networks to access, receive, or transmit data – sensitive or otherwise. Networks include GSM, 3G, and 4G; Bluetooth, 802.11-based wireless networks, and SMS and MMS texts. Each has its own security strengths and limitations, and each is a vector for remote exploits including data leakage or interception by malicious users – particularly if the mobile user isn't aware of the dangers or vulnerabilities.

- **Data Privacy and Security Concerns** – Mobile applications can take advantage of mobile-specific capabilities such as precise and continuous device location information (a weather app that uses device location to display local weather), physical sensor data (Touch ID), personal health metrics (Fitbit), and photos and audio about the device's user. The data collected by, stored in, and often transmitted by these mobile applications, though, raises privacy and security concerns if that information is exposed via these applications to unauthorized users.

## ➢ App Risks

Attackers can potentially use many different paths through your application to do harm to your business or organization. Each of these paths represents a risk that may, or may not, be serious enough to warrant attention. Sometimes these paths are trivial to find and exploit, and sometimes they are extremely difficult. Similarly, the harm that is caused may be of no consequence, or it may put you out of business. To determine the risk to your organization, you can evaluate the likelihood associated with each threat agent, attack vector, and security weakness and combine it with an estimate of the technical and business impact to your organization. Together, these factors determine your overall risk.

Risks.

Each organization is unique, and so are the threat actors for that organization, their goals, and the impact of any breach. If a public interest organization uses a content management system (CMS) for public information and a health system uses that same exact CMS for sensitive health records, the threat actors and business impacts can be very different for the same software. It is critical to understand the risk to your organization based on applicable threat agents and business impacts. Where possible, the names of the risks in the Top 10 are aligned with Common Weakness Enumeration (CWE) weaknesses to promote generally accepted naming conventions and to reduce confusion