UNIT-1

ATTACKS ON COMPUTERS AND COMPUTER SECURITY

INTRODUCTION

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

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

• **Computer Security** - generic name for the collection of tools designed to protect data and to thwart hackers

• **Network Security** - measures to protect data during their transmission

• **Internet Security** - measures to protect data during their transmission over a collection of interconnected networks

Need of Network Security:-

The network needs security against attackers and hackers. Network Security includes two basic securities. The first is the security of data information i.e. to protect the information from unauthorized access and loss. And the second is computer security i.e. to protect data and to thwart hackers. Here network security not only means security in a single network rather in any network or network of networks.

Now our need of network security has broken into two needs. One is the need of information security and other is the need of computer security.

On internet or any network of an organization, thousands of important information is exchanged daily. This information can be misused by attackers. The information security is needed for the following given reasons.

1. To protect the secret information users on the net only. No other person should see or access it.
2. To protect the information from unwanted editing, accidently or intentionally by unauthorized users.
3. To protect the information from loss and make it to be delivered to its destination properly.
4. To manage for acknowledgement of message received by any node in order to protect from denial by sender in specific situations. For example let a customer orders to purchase a few shares XYZ to the broader and denies for the order after two days as the rates go down.
5. To restrict a user to send some message to another user with name of a third one. For example a user X for his own interest makes a message containing some favorable instructions and sends it to user Y in such a manner that Y accepts the message as coming from Z, the manager of the organization.
6. To protect the message from unwanted delay in the transmission lines/route in order to deliver it to required destination in time, in case of urgency.
7. To protect the data from wandering the data packets or information packets in the network for infinitely long time and thus increasing congestion in the line in case destination machine fails to capture it because of some internal faults.

Principles of Security

There are several types of security attacks are performed by hackers in real life. To tackle those attacks, there are sets of security principles which will help us to understand and find the possible solutions to those problems which are caused by the attackers. There are six principles of cryptography and security.

1. **Confidentiality:** It specifies that only the sender and the authentic user should be able to access the data. No third party will access the data without authorization.

2. **Authentication:** It is the mechanism to establishing a secure (authentic) connection between sender and receiver. It is mainly used in electronic transactions, and network handshaking.

3. **Data Integrity:** When the sender sends a message to the recipient and in between the communication channel the contents of a message is lost. Then it says that the integrity of the message is lost.

4. **Non-Repudiation:** This refers to the situation when a user sends the data and at a later stage it denies.

5. **Access Control:** It defines as what to access and who has given the right to access the data.

6. **Availability:** It specifies that all resources are available to the legitimate users at all time.

Security Attacks

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

Passive Attacks

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

1. Release of message contents:-

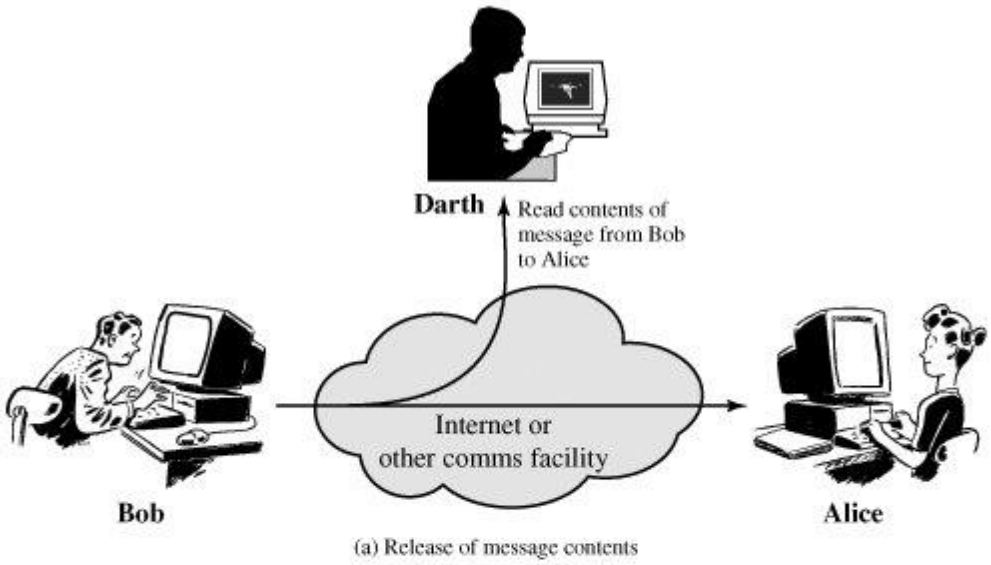
The **release of message contents** is easily understood. A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.

2. Traffic analysis:-

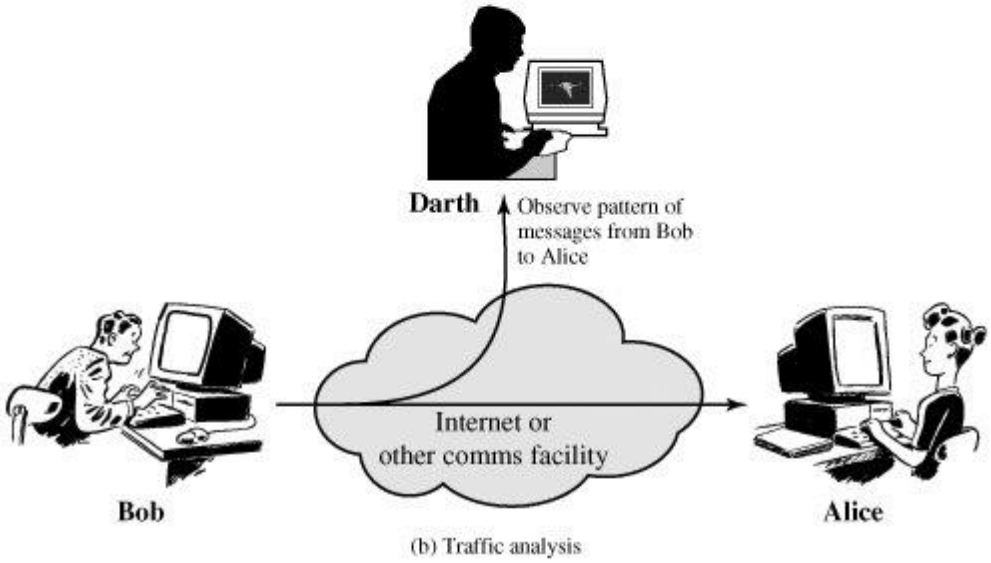
A second type of passive attack is traffic analysis. Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. Typically, the message traffic is not sent and received in an apparently normal fashion and the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern. However, it is feasible to prevent the success

of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.



(a) Release of message contents



(b) Traffic analysis

Active Attacks

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

1. Masquerade:-

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

2. Replay:-

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

3. Modification of messages:-

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

4. Denial of service:-

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

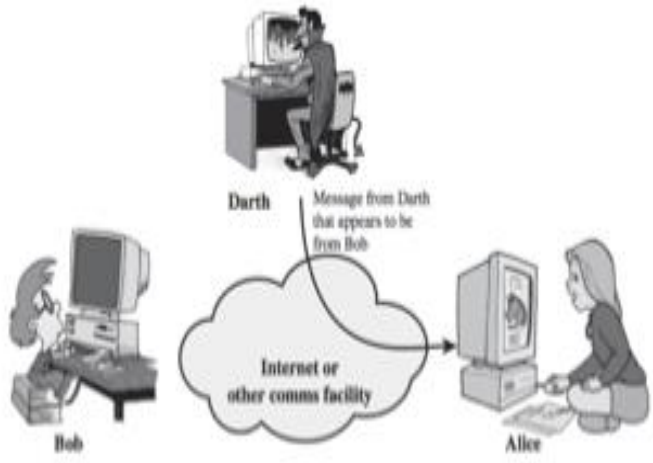


Figure 1.7 Masquerade

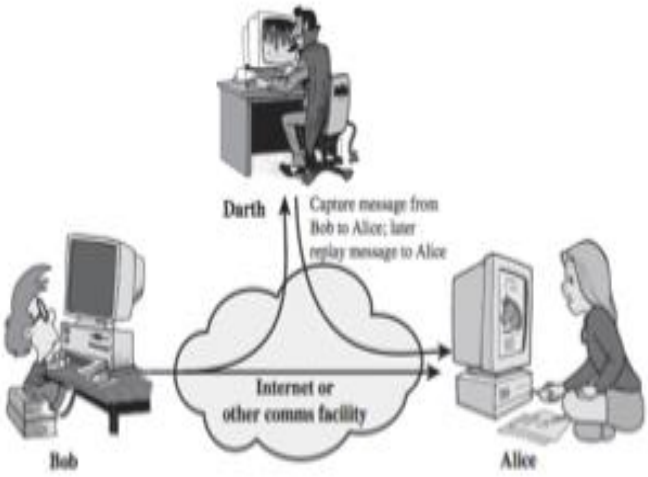


Figure 1.7 Replay

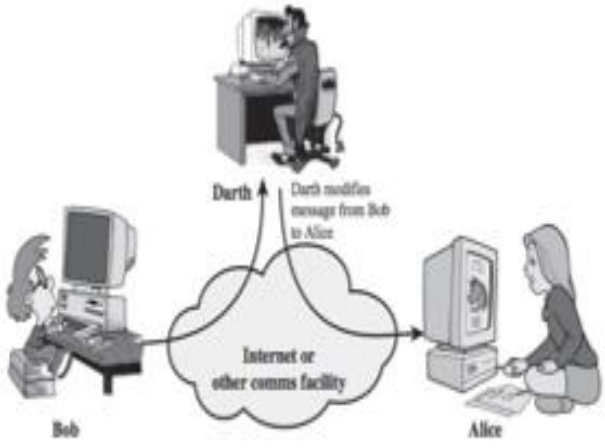


Figure 1.7 Modification of Messages

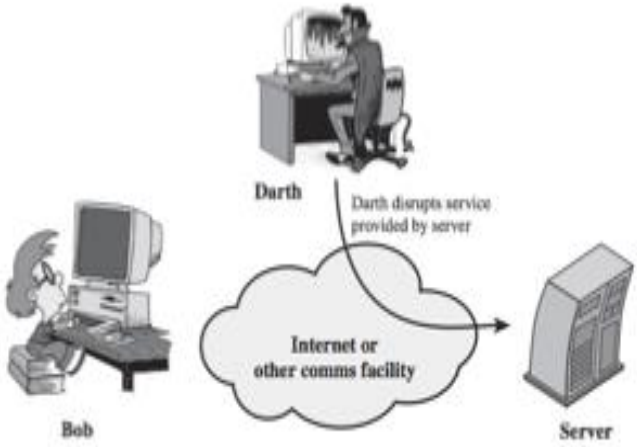


Figure 1.7 Denial of Service (DoS)

Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely, because of the wide variety of potential physical, software, and network vulnerabilities. Instead, the goal is to detect active attacks and to recover from any disruption or delays caused by them. If the detection has a deterrent effect, it may also contribute to prevention.

SECURITY SERVICES

A processing or communication service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.

1. AUTHENTICATION

The authentication service is concerned with assuring that a communication is Authentic, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an ongoing interaction, such as the connection of a terminal to a host, two aspects are involved.

Peer Entity Authentication

Used in association with a logical connection to provide confidence in the identity of the entities connected.

Data Origin Authentication

In a connectionless transfer, provides assurance that the source of received data is as claimed.

2. ACCESS CONTROL

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

3. DATA CONFIDENTIALITY

The protection of data from unauthorized disclosure. Confidentiality is the protection of transmitted data from passive attacks. With respect to the content of a data transmission, several levels of protection can be identified.

Connection Confidentiality: The protection of all user data on a connection.

Connectionless Confidentiality: The protection of all user data in a single data block.

Selective-Field confidentiality: The confidentiality of selected fields within the user data on a connection or in a single data block.

Traffic Flow Confidentiality: The protection of the information that might be derived from observation of traffic flows.

4. DATA INTEGRITY

The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).

Connection Integrity with Recovery

Provides for the integrity of all user data on a connection and detects any modification, insertion, deletion, or replay of any data within an entire data sequence, with recovery attempted.

Connection Integrity without Recovery: As above, but provides only detection without recovery.

Selective-Field Connection Integrity: Provides for the integrity of selected fields within the user data of a data block transferred over a connection and takes the form of determination of whether the selected fields have been modified, inserted, deleted, or replayed.

Connectionless Integrity: Provides for the integrity of a single connectionless data block and may take the form of detection of data modification. Additionally, a limited form of replay detection may be provided.

Selective-Field Connectionless Integrity: Provides for the integrity of selected fields within a single connectionless data block; takes the form of determination of whether the selected fields have been modified.

5. NONREPUDIATION

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

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

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

SECURITY MECHANISMS

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

SPECIFIC SECURITY MECHANISMS

May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.

Encipherment: The use of mathematical algorithms to transform data into a form that is not readily intelligible. The transformation and subsequent recovery of the data depend on an algorithm and zero or more encryption keys.

Digital Signature: Data appended to, or a cryptographic transformation of, a data unit that allows a recipient of the data unit to prove the source and integrity of the data unit and protect against forgery (e.g., by the recipient).

Access Control: A variety of mechanisms that enforce access rights to resources.

Data Integrity: A variety of mechanisms used to assure the integrity of a data unit or stream of data units

Authentication Exchange: A mechanism intended to ensure the identity of an entity by means of information exchange.

Traffic Padding: The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.

Routing Control: Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.

Notarization: The use of a trusted third party to assure certain properties of a data exchange.

PERVASIVE SECURITY MECHANISMS

Mechanisms that is not specific to any particular OSI security service or protocol layer.

Trusted Functionality: That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).

Security Label: The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.

Event Detection: Detection of security-relevant events.

Security Audit Trail: Data collected and potentially used to facilitate a security audit, which is an independent review and examination of system records and activities.

Security Recovery: Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.

A Model for Network Security

When we send our data from source side to destination side we have to use some transfer method like the internet or any other communication channel by which we are able to send our message. The two parties, who are the principals in this transaction, must cooperate for the exchange to take place. When the transfer of data happened from one source to another source some logical information channel is established between them by defining a route through the internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.

When we use the protocol for this logical information channel the main aspect security has come. Who may present a threat to confidentiality, authenticity, and so on? All the technique for providing security has to components:

1. A security-related transformation on the information to be sent.
2. Some secret information shared by the two principals and, it is hoped, unknown to the opponent.

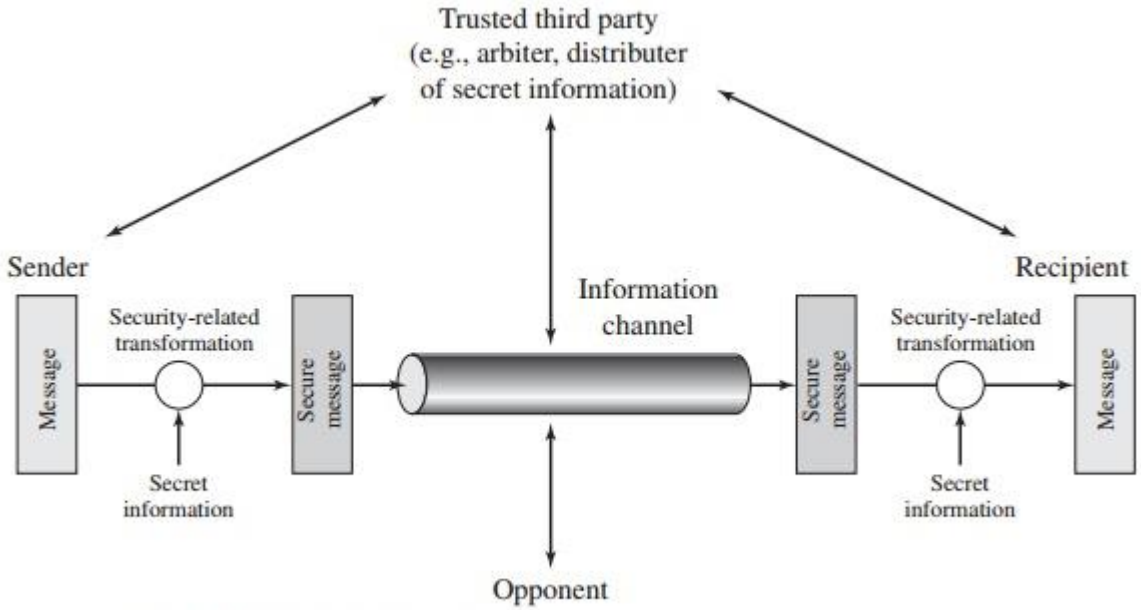


Figure 1.4 Model for Network Security

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 model shows that there are four basic tasks in designing a particular security service:

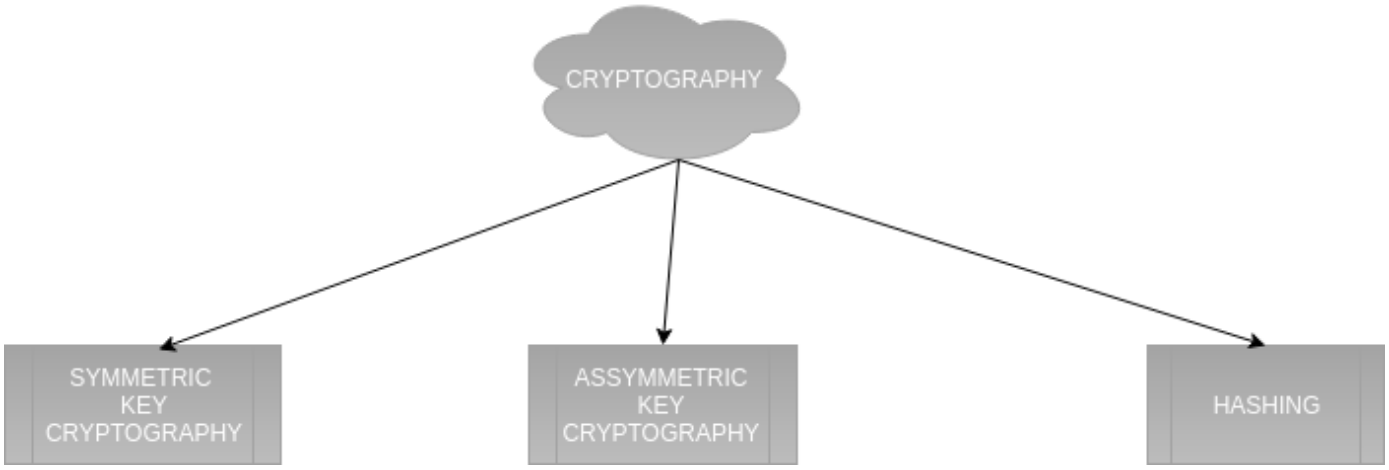
1. Design an algorithm for performing the security-related transformation.
2. Generate the secret information to be used with the algorithm.
3. Develop methods for the distribution and sharing of secret information.
4. Specify a protocol to be used by the two principals that make use of the security algorithm and the secret information to achieve a particular security service.

Cryptography

Cryptography

Cryptography is technique of securing information and communications through use of codes so that only those people for whom the information is intended can understand it and process it. Thus preventing unauthorized access to information. The prefix "crypt" means "hidden" and suffix graphy means "writing".

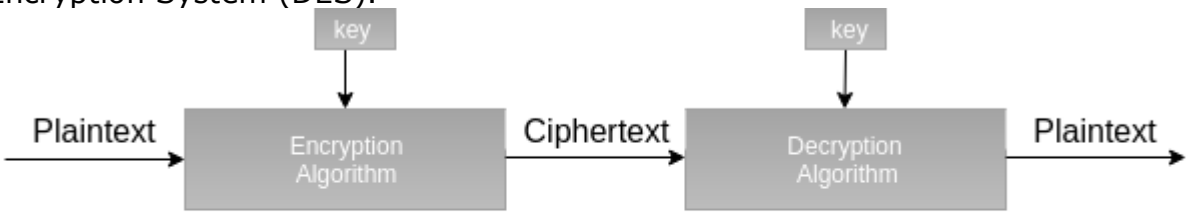
In Cryptography the techniques which are use to protect information are obtained from mathematical concepts and a set of rule based calculations known as algorithms to convert messages in ways that make it hard to decode it. These algorithms are used for cryptographic key generation, digital signing, and verification to protect data privacy, web browsing on internet and to protect confidential transactions such as credit card and debit card transactions.



Types Of Cryptography:

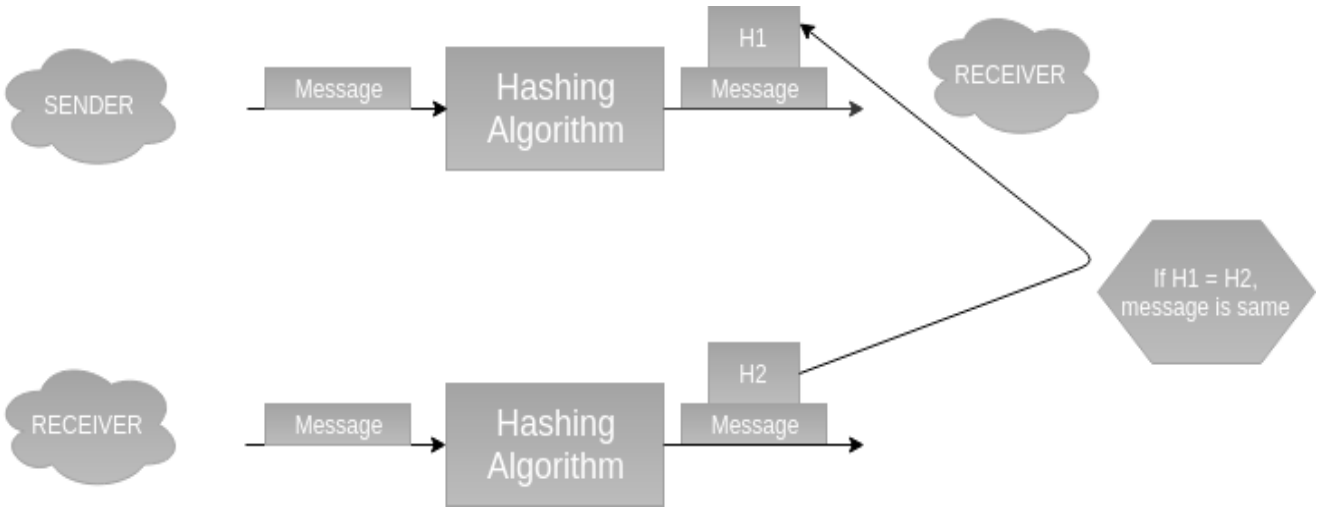
In general there are three types Of cryptography:

1. **Symmetric Key Cryptography:**
   It is an encryption system where the sender and receiver of message use a single common key to encrypt and decrypt messages. Symmetric Key Systems are faster and simpler but the problem is that sender and receiver have to somehow exchange key in a secure manner. The most popular symmetric key cryptography system is Data Encryption System (DES).

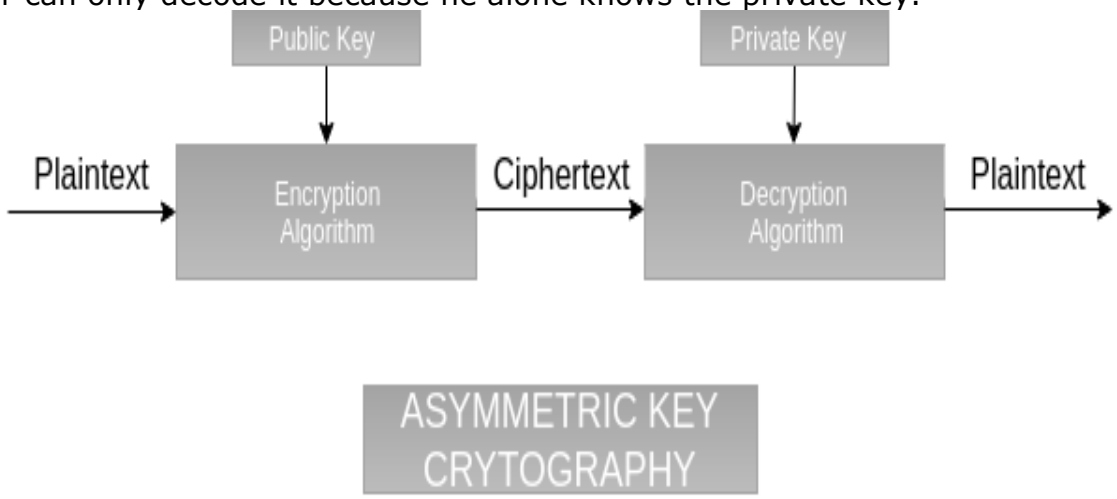


SYMMETRIC KEY CRYTOGRAPHY

2. **Hash Functions:**
   There is no usage of any key in this algorithm. A hash value with fixed length is calculated as per the plain text which makes it impossible for contents of plain text to be recovered. Many operating systems use hash functions to encrypt passwords.



3. **Asymmetric Key Cryptography:**
   Under this system a pair of keys is used to encrypt and decrypt information. A public key is used for encryption and a private key is used for decryption. Public key and Private Key are different. Even if the public key is known by everyone the intended receiver can only decode it because he alone knows the private key.



Symmetric Cipher Model

A symmetric encryption scheme has five ingredients.

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

Encryption algorithm: The encryption algorithm performs various substitutions and transformations on the plaintext.

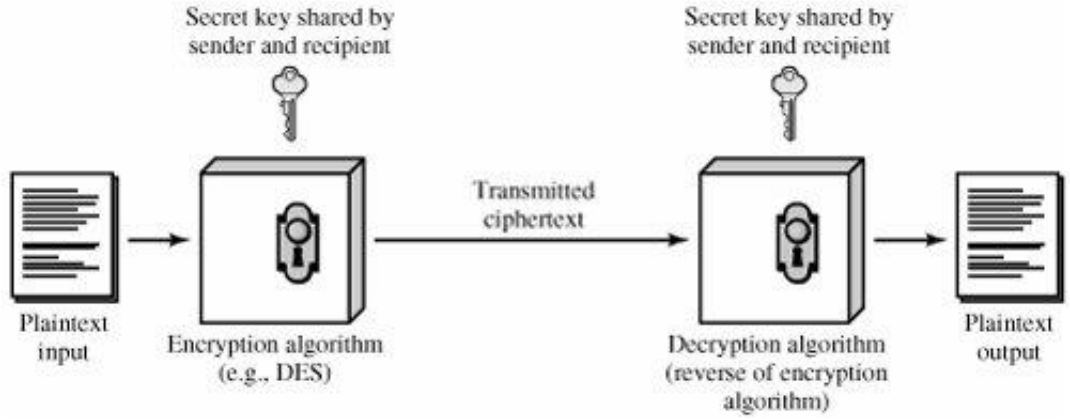
Secret key: The secret key is also input to the encryption algorithm. The key is a value independent of the plaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. The exact substitutions and transformations performed by the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

Cipher text: This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different cipher texts.

The cipher text is an apparently random stream of data and, as it stands, is unintelligible.

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

Figure Simplified Model of Conventional Encryption



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

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 an estimate.

CLASSICAL ENCRYPTION TECHNIQUES

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

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 cipher text bit patterns.

1. Caesar 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 let-ter standing three places further down the alphabet.
- For example,
  Plain text: meet me after the toga party
  Cipher text: PHHW PH DIWHU WKH WRJD SDUWB
- Note that the alphabet is wrapped around, so that the letter following Z is A. We can define the transformation by listing all possibilities, as follows:
  Plain: 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

Then the algorithm can be expressed as follows. For each plaintext letter *p*, substitute the cipher text letter *C*

$$C = E(3, p) = (p + 3) \bmod 26$$

A shift may be of any amount, so that the general Caesar algorithm is

$$C = E(k, p) = (p + k) \bmod 26$$

Where *k* takes on a value in the range 1 to 25.

The decryption algorithm is simply

$$p = D(k, C) = (C - k) \bmod 26$$

If it is known that a given cipher text is a Caesar cipher, then a brute-force cryptanalysis is easily performed and simply try all the 25 possible keys.

2. 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 pro-ceeding, we define the term *permutation*. 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.

If, instead, the "cipher" line can be any permutation of the 26 alphabetic characters, then there are 26! Or greater than $4 * 10^{26}$ 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 **monoalphabetic substitution cipher**, because a single cipher alphabet (mapping from plain alphabet to cipher alphabet) is used per message.

There is, however, another line of attack. If the cryptanalyst knows the nature of the plaintext (e.g., noncom pressed English text), then the analyst can exploit the regularities of the language. The cipher text to be solved is

UZQSOVUOHXMOPVGPOZPEVSGZWSZOPFPESXUDBMETSXAIZ

VUEPHZHMDZSHZOWSFPAPPDTSVPQUZWYMXUZUHSX

EPYEPOPDZSZUFPOMBZWPFUPZHMDJUDTMOHMQ

As a first step, the relative frequency of the letters can be determined and compared to a standard frequency distribution for English, such as is shown in Figure.

If the message were long enough, this technique alone might be sufficient, but because this is a relatively short message, we cannot expect an exact match. In any case, the relative frequencies of the letters in the cipher text (in percentages) are as following figure.

A powerful tool is to look at the frequency of two-letter combinations, known as **digrams**. A table similar to Figure 2.5 could be drawn up showing the relative frequency of digrams. The most common such digram is TH. In our cipher text, the most common digram is ZW, which appears three times. So we make the correspondence of Z with t and W with h. Then, by our earlier hypothesis, we can equate P with e. Now notice that the sequence ZWP appears in the cipher text, and we can translate that sequence as "the." This is the most frequent trigram (three-letter combination) in English, which seems to indicate that we are on the right track.

P 13.33	H 5.83	F 3.33	B 1.67	C 0.00
Z 11.67	D 5.00	W 3.33	G 1.67	K 0.00
S 8.33	E 5.00	Q 2.50	Y 1.67	L 0.00
U 8.33	V 4.17	T 2.50	I 0.83	N 0.00
O 7.50	X 4.17	A 1.67	J 0.83	R 0.00
M 6.67				

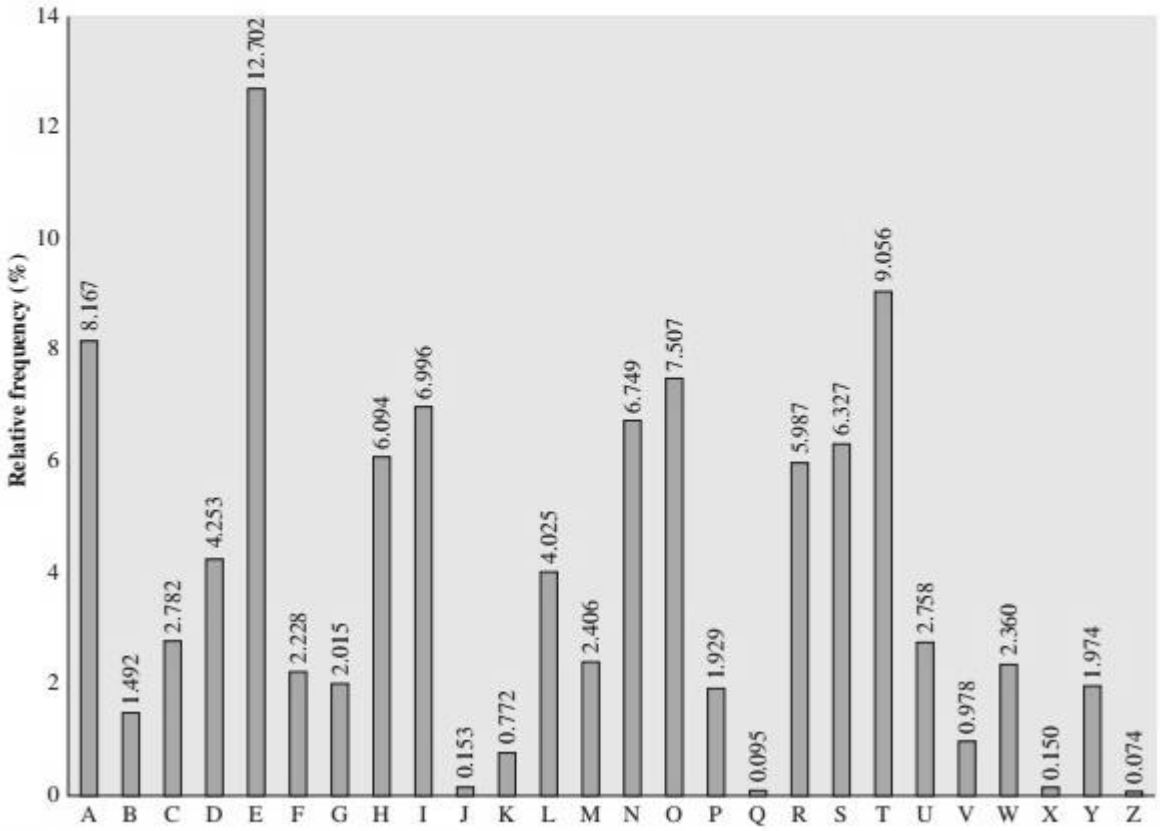


Figure 2.5 Relative Frequency of Letters in English Text

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. If the number of symbols assigned to each letter is proportional to the relative frequency of that letter, then single-letter frequency information is completely obliterated. The great mathematician **Carl Friedrich Gauss** believed that he had devised an unbreakable cipher using homo-phones. However, even with homophones, each element of plaintext affects only one element of cipher text, and multiple-letter patterns (e.g., digram frequencies) still survive in the cipher text, making cryptanalysis relatively straightforward.

3. Polyalphabetic ciphers

Polyalphabetic Cipher is a substitution cipher in which the cipher alphabet for the plain alphabet may be different at different places during the encryption process. The next two examples, **playfair and Vigenere Cipher are polyalphabetic ciphers**.

Playfair cipher:-

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

The Playfair algorithm is based on the use of a 5 x 5 matrix of letters constructed using a keyword. Here is an example, solved by Lord Peter Wimsey in Dorothy Sayers's *Have His Carcase:*

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 bot-tom, 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:

1. 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.**

2. 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.

3. 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.

4. Otherwise, each plaintext letter in a pair is replaced by the letter that lies in its own row and the column occupied by the other plaintext letter. Thus, hs becomes BP and ea becomes IM (or JM, as the encipherer wishes).

The Playfair cipher is a great advance over simple monoalphabetic ciphers. For one thing, whereas there are only 26 letters, there are 26 x 26 = 676 digrams, 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 much more 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.

Vigenere Cipher

This scheme of cipher uses a text string (say, a word) as a key, which is then used for doing a number of shifts on the plaintext.

For example, let's assume the key is 'point'. Each alphabet of the key is converted to its respective numeric value: In this case,

p → 16, o → 15, i → 9, n → 14, and t → 20.

Thus, the key is: 16 15 9 14 20.

Process of Vigenere Cipher:-

- The sender and the receiver decide on a key. Say 'point' is the key. Numeric representation of this key is '16 15 9 14 20'.
- The sender wants to encrypt the message, say 'attack from south east'. He will arrange plaintext and numeric key as follows –

a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14

- He now shifts each plaintext alphabet by the number written below it to create cipher text as shown below –

a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14
Q	I	C	O	W	A	U	A	C	G	I	D	D	H	B	U	P	B	H

- Here, each plaintext character has been shifted by a different amount – and that amount is determined by the key. The key must be less than or equal to the size of the message.
- For decryption, the receiver uses the same key and shifts received cipher text in reverse order to obtain the plaintext.

Q	I	C	O	W	A	U	A	C	G	I	D	D	H	B	U	P	B	H
16	15	9	14	20	16	15	9	14	20	16	15	9	14	20	16	15	9	14
a	t	t	a	c	k	f	r	o	m	s	o	u	t	h	e	a	s	t

Security Value:-

Vigenere Cipher was designed by tweaking the standard Caesar cipher to reduce the effectiveness of cryptanalysis on the cipher text and make a cryptosystem more robust. It is significantly **more secure than a regular Caesar Cipher**.

In the history, it was regularly used for protecting sensitive political and military information. It was referred to as the **unbreakable cipher** due to the difficulty it posed to the cryptanalysis.

4. Hill cipher

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

Hill cipher let us briefly review some terminology from linear algebra. In this discussion, we are concerned with matrix arithmetic modulo 26 and inverse matrix.

We define the inverse $\mathbf{M}^{-1}$ of a square matrix **M** by the equation $\mathbf{M}(\mathbf{M}^{-1}) = \mathbf{M}^{-1}\ \mathbf{M} = \mathbf{I}$, where **I** is the identity matrix. **I** is a square matrix that is all zeros except for ones along the main diagonal from upper left to lower right. The inverse of a matrix does not always exist, but when it does, it satisfies the preceding equation. For example,

$$\mathbf{A} = \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} \qquad \mathbf{A}^{-1} \bmod 26 = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

$$\mathbf{A}\mathbf{A}^{-1} = \begin{pmatrix} (5 \times 9) + (8 \times 1) & (5 \times 2) + (8 \times 15) \\ (17 \times 9) + (3 \times 1) & (17 \times 2) + (3 \times 15) \end{pmatrix}$$
$$= \begin{pmatrix} 53 & 130 \\ 156 & 79 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

To explain how the inverse of a matrix is computed, we begin by with the con-cept of determinant. For any square matrix (m x m), the **determinant** equals the sum of all the products that can be formed by taking exactly one element from each row and exactly one element from each column, with certain of the product terms pre-ceded by a minus sign. For a 2 x 2 matrix,

$$\begin{pmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{pmatrix}$$

the determinant is $k_{11}k_{22} - k_{12}k_{21}$. For a 3×3 matrix, the value of the determinant is $k_{11}k_{22}k_{33} + k_{21}k_{32}k_{13} + k_{31}k_{12}k_{23} - k_{31}k_{22}k_{13} - k_{21}k_{12}k_{33} - k_{11}k_{32}k_{23}$. If a square matrix **A** has a nonzero determinant, then the inverse of the matrix is

computed as $[\mathbf{A}^{-1}]_{ij} = (\det \mathbf{A})^{-1}(-1)^{i+j}(D_{ji})$, where (D_{ji}) is the subdeterminant formed by deleting the jth row and the ith column of $\mathbf{A}$, det($\mathbf{A}$) is the determinant of $\mathbf{A}$, and $(\det \mathbf{A})^{-1}$ is the multiplicative inverse of (det $\mathbf{A}$) mod 26.

Continuing our example,

$$\det \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix} = (5 \times 3) - (8 \times 17) = -121 \bmod 26 = 9$$

We can show that $9^{-1} \bmod 26 = 3$, because $9 \times 3 = 27 \bmod 26 = 1$ (see Chapter 4 or Appendix E). Therefore, we compute the inverse of $\mathbf{A}$ as

$$\mathbf{A} = \begin{pmatrix} 5 & 8 \\ 17 & 3 \end{pmatrix}$$

$$\mathbf{A}^{-1} \bmod 26 = 3\begin{pmatrix} 3 & -8 \\ -17 & 5 \end{pmatrix} = 3\begin{pmatrix} 3 & 18 \\ 9 & 5 \end{pmatrix} = \begin{pmatrix} 9 & 54 \\ 27 & 15 \end{pmatrix} = \begin{pmatrix} 9 & 2 \\ 1 & 15 \end{pmatrix}$$

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 can be described as

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

This can be expressed in terms of row vectors and matrices:[7]

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

or

$$\mathbf{C} = \mathbf{PK} \bmod 26$$

Where **C** and **P** are row vectors of length 3 representing the plaintext and cipher text, and **K** is a 3 * 3 matrix representing the encryption key. Operations are performed mod 26. For example, consider the plaintext "paymoremoney" and use the encryption key

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

The first three letters of the plaintext are represented by the vector (15 0 24). Then (15 0 24)**K** = (303 303 531) mod 26 = (17 17 11) = RRL. Continuing in this fashion, the cipher text for the entire plaintext is RRLMWBKASPDH.

Decryption requires using the inverse of the matrix **K**. We can compute det **K** = 23, and therefore, (det **K**)$^{-1}$ mod 26 = 17. We can then compute the inverse as

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

This is demonstrated as

$$\begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \begin{pmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{pmatrix} = \begin{pmatrix} 443 & 442 & 442 \\ 858 & 495 & 780 \\ 494 & 52 & 365 \end{pmatrix} \bmod 26 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

It is easily seen that if the matrix $\mathbf{K}^{-1}$ is applied to the cipher text, then the plaintext is recovered.

In general terms, the Hill system can be expressed as

$\mathbf{C} = E(\mathbf{K}, \mathbf{P}) = \mathbf{PK} \bmod 26$

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

As with Playfair, the strength of the Hill cipher is that it completely hides single-letter frequencies. Indeed, with Hill, the use of a larger matrix hides more frequency information. Thus, a 3 * 3 Hill cipher hides not only single-letter but also two-letter frequency information.

5. 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. His system works on binary data (bits) rather than letters. The system can be expressed succinctly as follows

$$c_i = p_i \oplus k_i$$

where

p_i = ith binary digit of plaintext
k_i = ith binary digit of key
c_i = ith binary digit of ciphertext
$\oplus$ = exclusive-or (XOR) operation

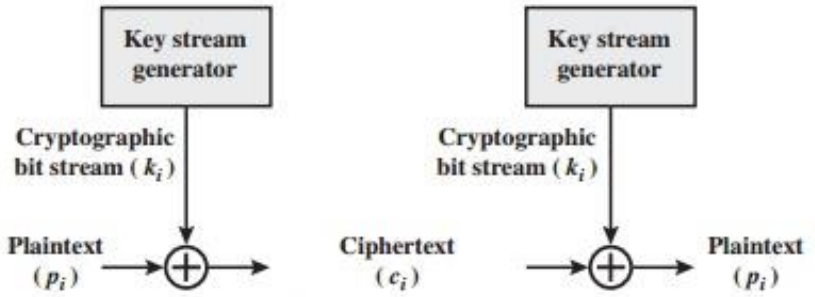


Figure 2.7 Vernam Cipher

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

$$p_i = c_i \oplus k_i$$

The essence of this technique is the means of construction of the key. Vernam proposed the use of a running loop of tape that eventually repeated the key, so that in fact the system worked with a very long but repeating keyword. Although such a scheme, with a long key, presents formidable cryptanalytic diffi-culties, it can be broken with sufficient ciphertext, the use of known or probable plaintext sequences, or both.

6. One-Time Pad

An Army Signal Corp officer, Joseph Mauborgne, proposed an improvement to the Vernam cipher that yields the ultimate in security. Mauborgne suggested using a random key that is as long as the message, so that the key need not be repeated. In addition, the key is to be used to encrypt and decrypt a single message, and then is discarded. Each new message requires a new key of the same length as the new message. Such a scheme, known as a **one-time pad**, is unbreakable. It produces random output that bears no statistical relationship to the plaintext. Because the cipher text contains no information whatsoever about the plaintext, there is simply no way to break the code.

Suppose that we are using a Vigenère scheme with 27 characters in which the twenty-seventh character is the space character, but with a one-time key that is as long as the message. Consider the cipher text

ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

We now show two different decryptions using two different keys:

ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

key: *pxlmvmsydofuyrvzwc tnlebnecvgdupahfzzlmnyih*

plaintext: mr mustard with the candlestick in the hall
ciphertext: ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

key: *mfugpmiydgaxgoufhkllmhsqdqogtewbqfgyovuhwt*

plaintext: miss scarlet with the knife in the library

Suppose that a cryptanalyst had managed to find these two keys. Two plausible plaintexts are produced. If the actual key were produced in a truly random fashion, then the cryptanalyst cannot say that one of these two keys is more likely than the other. Thus, there is no way to decide which key is correct and there-fore which plaintext is correct.

In fact, given any plaintext of equal length to the ciphertext, there is a key that produces that plaintext. Therefore, if you did an exhaustive search of all possible keys, you would end up with many legible plaintexts, with no way of knowing which the intended plaintext was. Therefore, the code is unbreakable.

The security of the one-time pad is entirely due to the randomness of the key. If the stream of characters that constitute the key is truly random, then the stream of characters that constitute the cipher text will be truly random. Thus, there are no patterns or regularities that a cryptanalyst can use to attack the cipher text.

The one-time pad offers complete security but, in practice, has two fundamental difficulties:

1. There is the practical problem of making large quantities of random keys. Any heavily used system might require millions of random characters on a regular basis. Supplying truly random characters in this volume is a significant task.

2. Even more daunting is the problem of key distribution and protection. For every message to be sent, a key of equal length is needed by both sender and receiver. Thus, a mammoth key distribution problem exists.

Because of these difficulties, the one-time pad is of limited utility and is useful primarily for low-bandwidth channels requiring very high security.

The one-time pad is the only cryptosystem that exhibits what is referred to as *perfect secrecy*.

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.

Transposition Techniques are based on the permutation of the plain-text instead of substitution.

1) Rail-Fence Technique

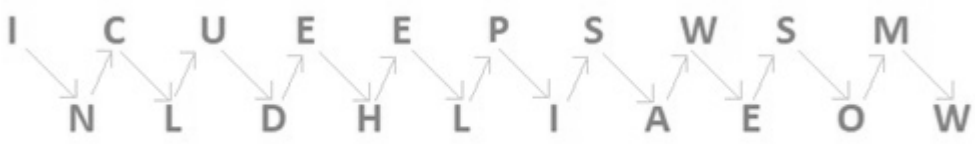
This technique is a type of Transposition technique and does is write the plain text as a sequence of diagonals and changing the order according to each row.

It uses a simple algorithm,

1. Writing down the plaintext message into a sequence of diagonals.
2. Row-wise writing the plain-text written from above step.

Example,

Let's say, we take an example of **"INCLUDEHELP IS AWESOME".**



So the Cipher-text are, **ICUEEPSWSMNLDHLIAEOW**.
First, we write the message in a zigzag manner then read it out direct row-wise to change it to cipher-text.
Now as we can see, Rail-Fence Technique is very to break by any cryptanalyst.

2) Columnar Transition Technique

A. Basic Technique
It is a slight variation to the Rail-fence technique, let's see its algorithm:

1. In a rectangle of pre-defined size, write the plain-text message row by row.
2. Read the plain message in random order in a column-wise fashion. It can be any order such as 2, 1, 3 etc.
3. Thus Cipher-text is obtained.

Example:
Original message: **"INCLUDEHELP IS AWESOME"**.
Now we apply the above algorithm and create the rectangle of 4 columns (we decide to make a rectangle with four columns it can be any number.)

Column 1	Column 2	Column 3	Column 4
I	N	C	L
U	D	E	H
E	L	P	I
S	A	W	E
S	O	M	E

Now let's decide on an order for the column as 4, 1, 3 and 2 and now we will read the text in column-wise.
Cipher-text: **LHIEEIUESSCEPWMNDLAO**

B. Columnar Technique with multiple rounds

In this method, we again change the chipper text we received from a Basic technique that is in round 1 and again follows the same procedure for the cipher-text from round 1.

Algorithm:

1. In a rectangle of pre-defined size, write the plain-text message row by row.
2. Read the plain message in random order in a column-wise fashion. It can be any order such as 2, 1, 3 etc.
3. Thus, Cipher-text of round 1 is obtained.
4. Repeat from step 1 to 3.

Example:

Original message: **"INCLUDEHELP IS AWESOME"**.

Now we apply the above algorithm and create the rectangle of 4 column (we decide to make a rectangle with four column it can be any number.)

Column 1	Column 2	Column 3	Column 4
I	N	C	L
U	D	E	H
E	L	P	I
S	A	W	E
S	O	M	E

Now let's decide on an order for the column as 4, 1, 3 and 2 and now we will read the text in column-wise.

Cipher-text of round 1: **LHIEEIUESSCEPWMNDLAO**

Round 2:

Column 1	Column 2	Column 3	Column 4
L	H	I	E
E	I	U	E
S	S	C	E
P	W	M	N
D	L	A	O

Now, we decide to go with a previous order that is 4,1,3,2.

Cipher-text: **EEENLESPICUMHISW**

These multi-round columnar techniques are harder to crack as compared to methods seen earlier.

STEGANOGRAPHY

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

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

e.g.,

(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.

Typewriter correction ribbon: Used between 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.

Key Range and Key Size

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

• If the key is found, the attacker can get original plaintext message. In the brute force attack, every possible key in the key-range is tried, until we get the right key.

• In the best case, the right key is found in the first attempt, in the worst case, the key is found in the last attempt. On an average, the right key is found after trying half of the possible keys in the key-range. Therefore by expanding the key range to a large extent, longer it will take for an attacker to find the key using brute-force attack.

• The concept of key range leads to the principle of key size. The strength of a cryptographic key is measured with the key size

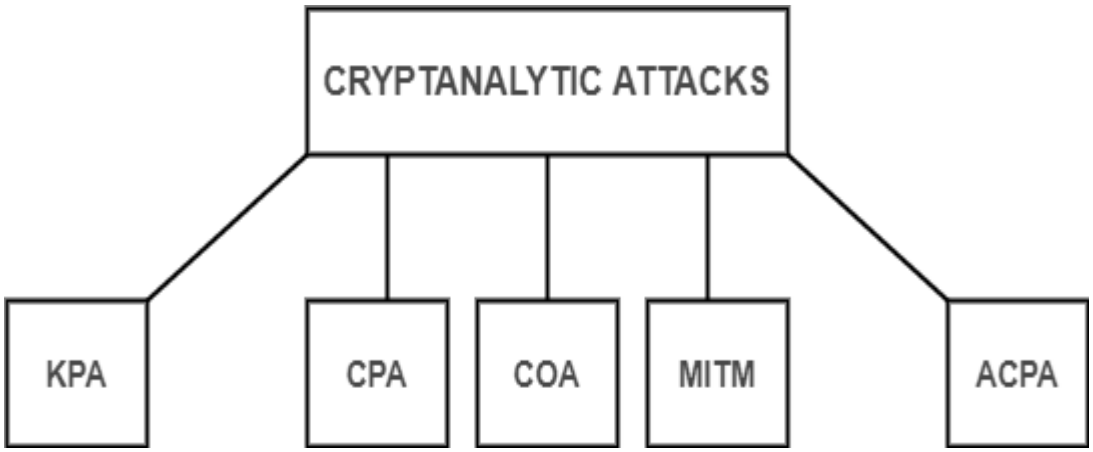
• Key size is measured in bits and is represented using binary number system. Thus if the key range from 0 to 8, then the key size is 3 bits or in other words we can say if the size is bits then the key range is 0 to 256. Key size may be varying, depending upon the applications and the cryptographic algorithm being used, it can be 40 bits, 56 bits, 128 bits & so on. In order to protect the cipher-text against the brute-force attack, the key-size should be such that the attacker cannot crack it within a specified amount of time.

• From a practical viewpoint, a 40-bit key takes about 3 hours to crack, however a 41-bit key would take 6 hours and 42-bit key would take 12 hours & so on. This means every additional bit doubles the amount of time required to crack the key. We can assume that 128 bit key is quite safe, considering the capabilities of today's computers. However as the computing power and techniques improve, these numbers will change in future.

Possible types of attacks

Cryptology has two parts namely, **Cryptography** which focuses on creating secret codes and **Cryptanalysis** which is the study of the cryptographic algorithm and the breaking of those secret codes. The person practicing Cryptanalysis is called a **Cryptanalyst**. It helps us to better understand the cryptosystems and also helps us improve the system by finding any weak point and thus work on the algorithm to create a more secure secret code. For example, a Cryptanalyst might try to decipher a cipher text to derive the plaintext. It can help us to deduce the plaintext or the encryption key.

To determine the weak points of a cryptographic system, it is important to attack the system. These attacks are called **Cryptanalytic attacks.** The attacks rely on nature of the algorithm and also knowledge of the general characteristics of the plaintext, i.e., plaintext can be a regular document written in English or it can be a code written in Java. Therefore, nature of the plaintext should be known before trying to use the attacks.



- **Known-Plaintext Analysis (KPA):**
  In this type of attack, some plaintext-cipher text pairs are already known. Attacker maps them in order to find the encryption key. This attack is easier to use as a lot of information is already available.
- **Chosen-Plaintext Analysis (CPA):**
  In this type of attack, the attacker chooses random plaintexts and obtains the corresponding cipher texts and tries to find the encryption key. Its very simple to implement like KPA but the success rate is quite low.

- **Cipher text-Only Analysis (COA):**
  In this type of attack, only some cipher-text is known and the attacker tries to find the corresponding encryption key and plaintext. Its the hardest to implement but is the most probable attack as only cipher text is required.
- **Man-In-The-Middle (MITM) attack:**
  In this type of attack, attacker intercepts the message/key between two communicating parties through a secured channel.
- **Adaptive Chosen-Plaintext Analysis (ACPA):**
  This attack is similar CPA. Here, the attacker requests the cipher texts of additional plaintexts after they have cipher texts for some texts.

****

UNIT-2

SYMMETRIC KEY CIPHERS

1. Stream Cipher

In stream cipher, one byte is encrypted at a time while in block cipher ~128 bits are encrypted at a time.

Initially, a key (k) will be supplied as input to pseudorandom bit generator and then it produces a random 8-bit output which is treated as key stream.

The resulted key stream will be of size 1 byte, i.e., 8 bits.

1. Stream Cipher follows the sequence of pseudorandom number stream.
2. One of the benefits of following stream cipher is to make cryptanalysis more difficult, so the number of bits chosen in the Key stream must be long in order to make cryptanalysis more difficult.
3. By making the key longer it is also safe against brute force attacks.
4. The longer the key the stronger security is achieved, preventing any attack.
5. Key stream can be designed more efficiently by including more number of 1s and 0s, for making cryptanalysis more difficult.
6. Considerable benefit of a stream cipher is, it requires few lines of code compared to block cipher.

Encryption:

For Encryption,

- Plain Text and Key stream produces Cipher Text (Same key stream will be used for decryption.).
- The Plaintext will undergo XOR operation with key stream bit-by-bit and produces the Cipher Text.

Example –

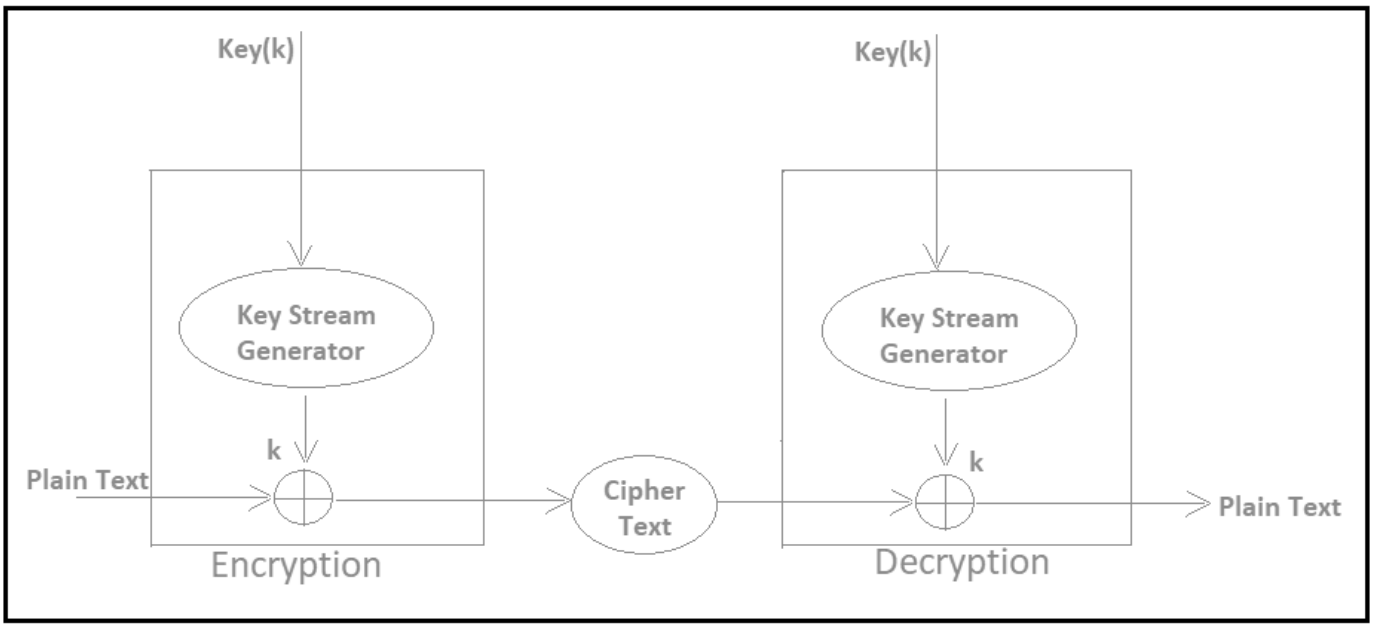
Plain Text: 10011001

Key stream: 11000011

``````````````````````

Cipher Text: 01011010

**Decryption:**

For Decryption,

- Cipher Text and Key stream gives the original Plain Text (Same key stream will be used for encryption.).
- The Cipher text will undergo XOR operation with keystream bit-by-bit and produces the actual Plain Text.

**Example –**

Cipher Text: 01011010

Key stream: 11000011

``````````````````````

Plain Text: 10011001

Decryption is just the reverse process of Encryption i.e. performing XOR with Cipher Text.



2. Block cipher

Block cipher is an encryption method which divides the plain text into blocks of fixed size. Each block has an equal number of bits. At a time, block cipher operates only on one block of plain text and applies key on it to produce the corresponding block of cipher text.

While decryption also only one block of cipher text is operated to produce its corresponding plain text. Data Encryption Standard (DES) is the best example of it.

DES divides the plain text into the number of blocks, each of 64-bit. DES operates on one block of plain text at a time. Key of 56-bit is applied to each block of plain text to produce its corresponding cipher text of 64-bit.

During decryption also only one block of ciphertext is operated at a time to produce its corresponding block plain text. In DES the decryption algorithm is the same as the encryption one.

On an all the block cipher operates on a block of bits at a time instead of one bit a time. Operating bit by bit is a very time-consuming process and as block cipher is a computer-based cryptographic algorithm it needs to be fast. That's why operating one block of bits at a time makes it faster as compared to a stream cipher.

But there was a limitation in the block cipher as it would generate the same cipher text for the repeating text pattern in plain text. However, this limitation was resolved by implementing **chaining** in the block cipher.

3. Block Cipher Design Principles

Block ciphers are built in the Feistel cipher structure. Block cipher has a specific number of rounds and keys for generating cipher text. For defining the complexity level of an algorithm few design principles are to be considered.
These are explained as following below:

1. **Number of Rounds –**
   The number of Rounds is regularly considered in design criteria, it just reflects the number of rounds to be suitable for an algorithm to make it more complex, and in DES we have 16 rounds ensuring it to be more secure while in AES we have 10 rounds which make it more secure.
2. **Design of function F –**
   The core part of the Feistel Block cipher structure is the Round Function. The complexity of cryptanalysis can be derived from the Round function i.e. the increasing level of complexity for the round function would be greatly contributing to an increase in complexity.
   To increase the complexity of the round function, the avalanche effect is also included in the round function, as the change of a single bit in plain text would produce a mischievous output due to the presence of avalanche effect.
3. **Key schedule algorithm –**
   In Feistel Block cipher structure, each round would generate a sub-key for increasing the complexity of cryptanalysis. The Avalanche effect makes it more complex in deriving sub-key. Decryption must be done very carefully to get the actual output as the avalanche effect is present in it.

4. Feistel Block cipher structure

Feistel Cipher is not a specific scheme of block cipher. It is a design model from which many different block ciphers are derived. DES is just one example of a Feistel Cipher. A cryptographic system based on Feistel cipher structure uses the same algorithm for both encryption and decryption.

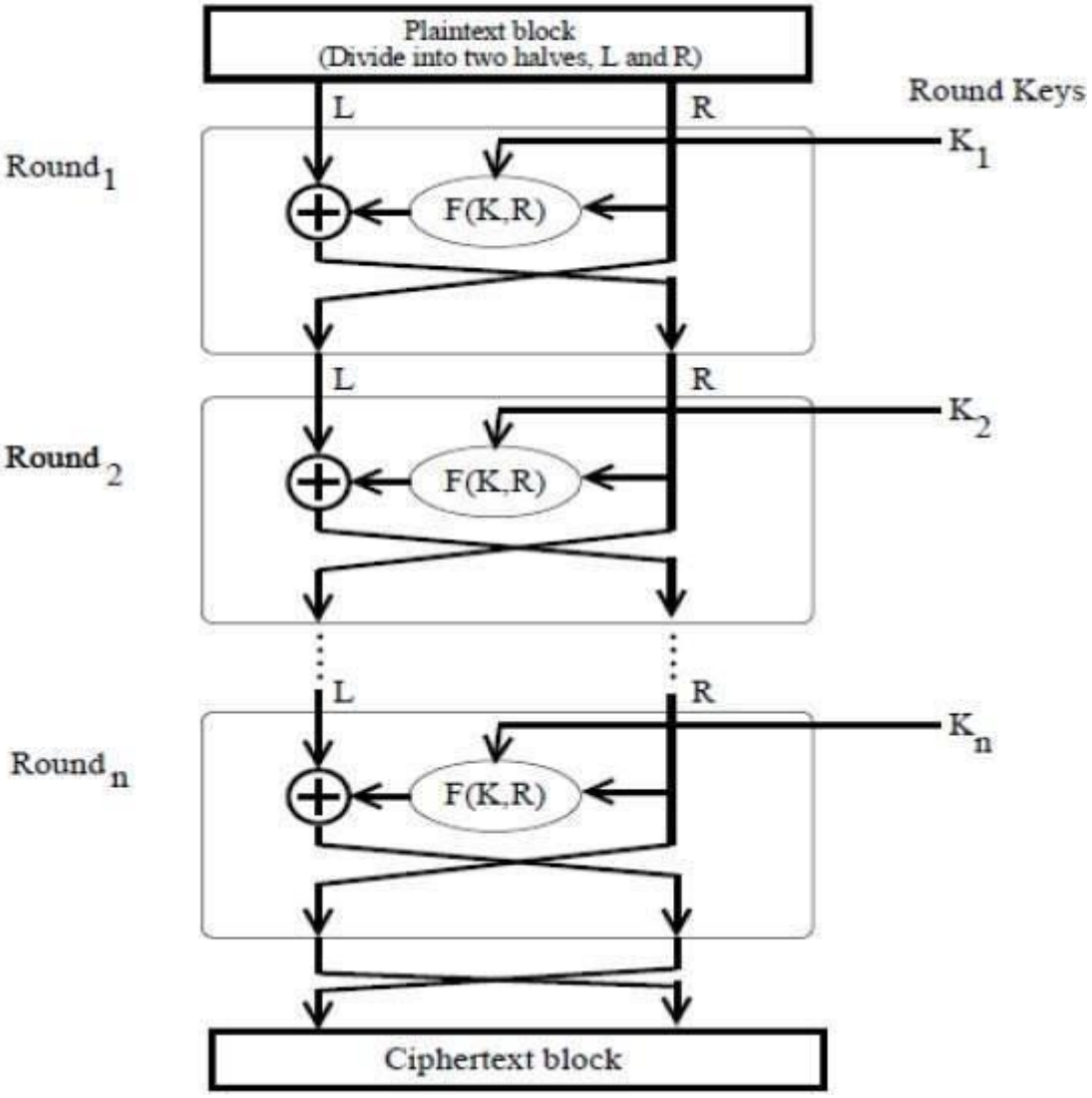
Encryption Process

The encryption process uses the Feistel structure consisting multiple rounds of processing of the plaintext, each round consisting of a "substitution" step followed by a permutation step.

- The input block to each round is divided into two halves that can be denoted as L and R for the left half and the right half.
- In each round, the right half of the block, R, goes through unchanged. But the left half, L, goes through an operation that depends on R and the encryption key. First, we apply an encrypting function 'f' that takes two input – the key K and R. The function produces the output f(R, K). Then, we XOR the output of the mathematical function with L.
- In real implementation of the Feistel Cipher, such as DES, instead of using the whole encryption key during each round, a round-dependent key (a sub key) is derived from the encryption key. This means that each round uses a different key, although all these sub keys are related to the original key.
- The permutation step at the end of each round swaps the modified L and unmodified R. Therefore, the L for the next round would be R of the current round. And R for the next round be the output L of the current round.
- Above substitution and permutation steps form a 'round'. The number of rounds are specified by the algorithm design.

- Once the last round is completed then the two sub blocks, 'R' and 'L' are concatenated in this order to form the ciphertext block.

Feistel Structure is shown in the following illustration –



The difficult part of designing a Feistel Cipher is selection of round function 'f'. In order to be unbreakable scheme, this function needs to have several important properties that are beyond the scope of our discussion.

Decryption Process

The process of decryption in Feistel cipher is almost similar. Instead of starting with a block of plaintext, the cipher text block is fed into the start of the Feistel structure and then the process thereafter is exactly the same as described in the given illustration.

The process is said to be almost similar and not exactly same. In the case of decryption, the only difference is that the sub keys used in encryption are used in the reverse order.

The final swapping of 'L' and 'R' in last step of the Feistel Cipher is essential. If these are not swapped then the resulting cipher text could not be decrypted using the same algorithm.

Number of Rounds

The number of rounds used in a Feistel Cipher depends on desired security from the system. More number of rounds provides more secure system. But at the same time, more rounds mean the inefficient slow encryption and decryption processes. Number of rounds in the systems thus depend upon efficiency–security tradeoff.

5. Difference between Block Cipher and Stream Cipher

S.NO	Block Cipher	Stream Cipher
1.	Block Cipher Converts the plain text into cipher text by taking plain text's block at a time.	Stream Cipher Converts the plain text into cipher text by taking 1 byte of plain text at a time.
2.	Block cipher uses either 64 bits or more than 64 bits.	While stream cipher uses 8 bits.
3.	The complexity of block cipher is simple.	While stream cipher is more complex.
4.	Block cipher Uses confusion as well as diffusion.	While stream cipher uses only confusion.
5.	In block cipher, reverse encrypted text is hard.	While in-stream cipher, reverse encrypted text is easy.
6.	The algorithm modes which are used in block cipher are ECB (Electronic Code Book) and CBC (Cipher Block Chaining).	The algorithm modes which are used in stream cipher are CFB (Cipher Feedback) and OFB (Output Feedback).
7.	Block cipher works on transposition techniques like rail-fence technique, columnar transposition technique, etc.	While stream cipher works on substitution techniques like Caesar cipher, Polygram substitution cipher, etc.
8.	Block cipher is slow as compared to a stream cipher.	While stream cipher is fast in comparison to block cipher.

6. Block Cipher modes of Operation

Encryption algorithms are divided into two categories based on the input type, as a block cipher and stream cipher. **Block cipher** is an encryption algorithm that takes a fixed size of input say *b* bits and produces a cipher text of *b* bits again. If the input is larger than *b* bits it can be divided further. For different applications and uses, there are several modes of operations for a block cipher.

Electronic Code Book (ECB) –

Electronic code book is the easiest block cipher mode of functioning. It is easier because of direct encryption of each block of input plaintext and output is in form of blocks of encrypted cipher text. Generally, if a message is larger than *b* bits in size, it can be broken down into a bunch of blocks and the procedure is repeated.

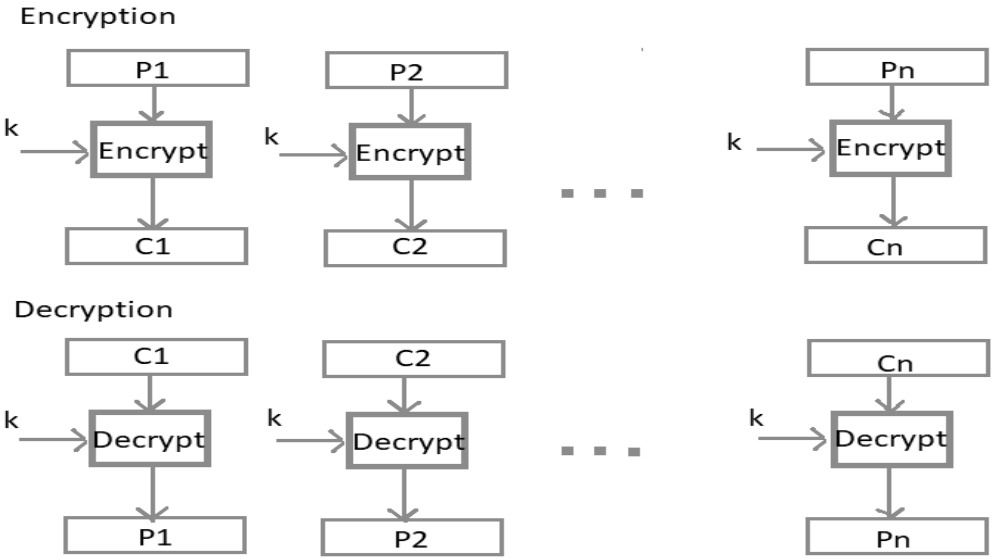
Advantages of using ECB –

- Parallel encryption of blocks of bits is possible, thus it is a faster way of encryption.
- Simple way of the block cipher.

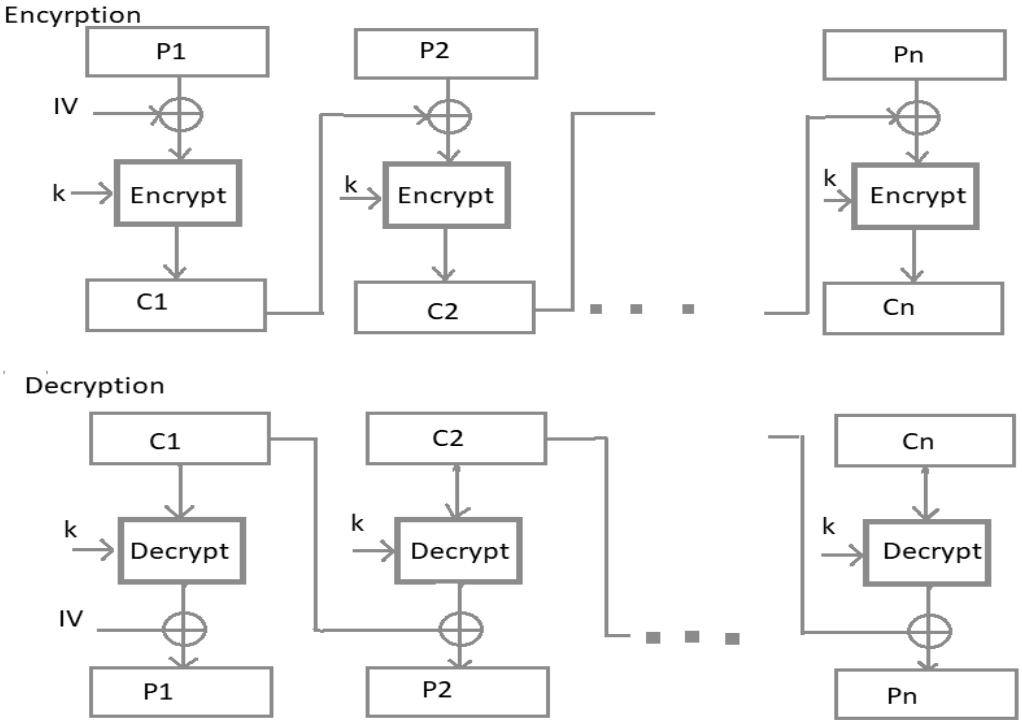
Disadvantages of using ECB –

- Prone to cryptanalysis since there is a direct relationship between plaintext and cipher text.

Procedure of ECB is illustrated below:



Cipher Block Chaining (CBC) –
Cipher block chaining or CBC is an advancement made on ECB since ECB compromises some security requirements. In CBC, the previous cipher block is given as input to the next encryption algorithm after XOR with the original plaintext block. In a nutshell here, a cipher block is produced by encrypting an XOR output of the previous cipher block and present plaintext block.
The process is illustrated here:



Advantages of CBC –

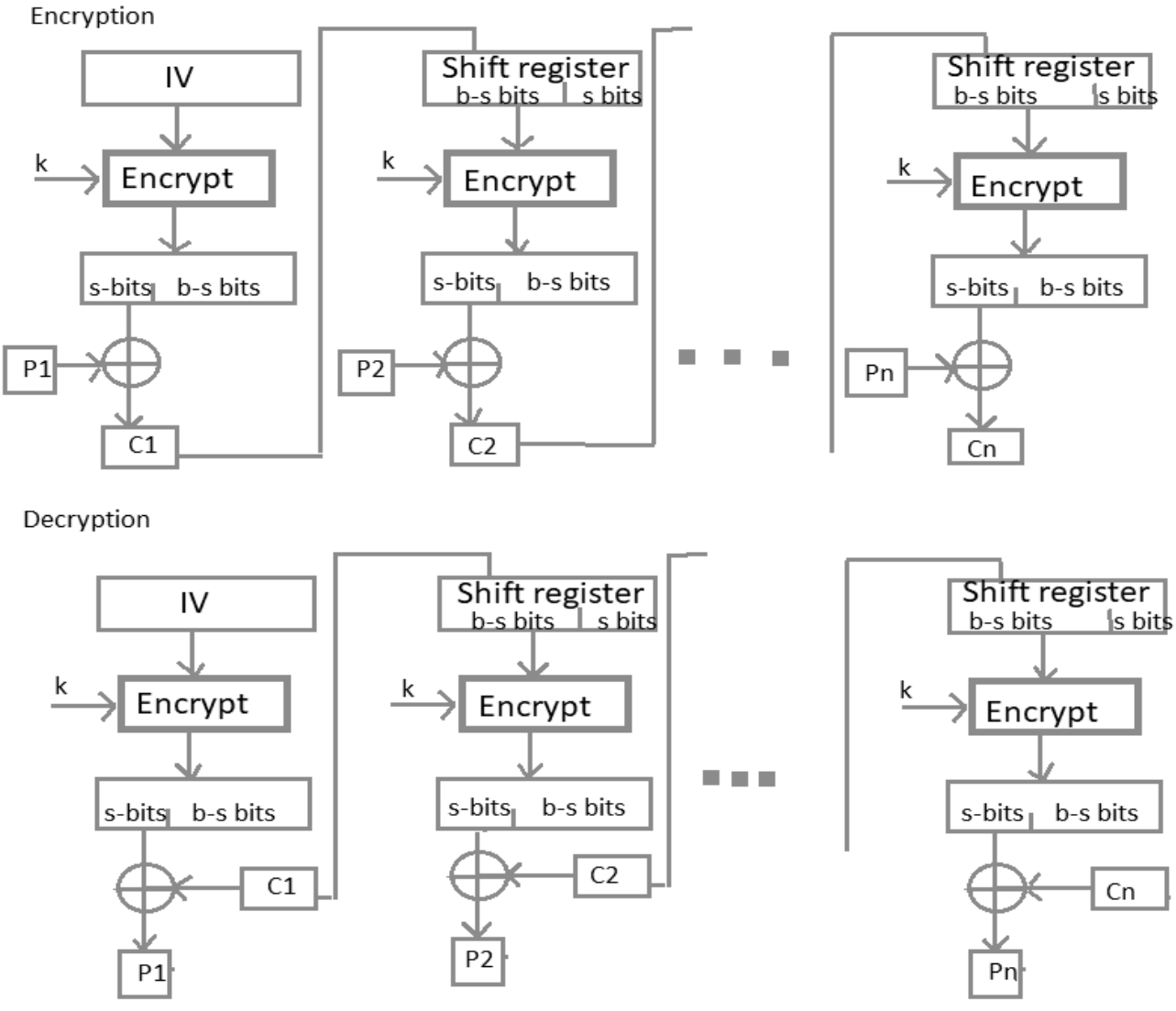
- CBC works well for input greater than *b* bits.
- CBC is a good authentication mechanism.
- Better resistive nature towards cryptanalysis than ECB.

Disadvantages of CBC –

- Parallel encryption is not possible since every encryption requires a previous cipher.

Cipher Feedback Mode (CFB) –

In this mode the cipher is given as feedback to the next block of encryption with some new specifications: first, an initial vector IV is used for first encryption and output bits are divided as a set of *s* and *b-s* bits. The left-hand side *s* bits are selected along with plaintext bits to which an XOR operation is applied. The result is given as input to a shift register having b-s bits to lhs, s bits to rhs and the process continues. The encryption and decryption process for the same is shown below; both of them use encryption algorithms.

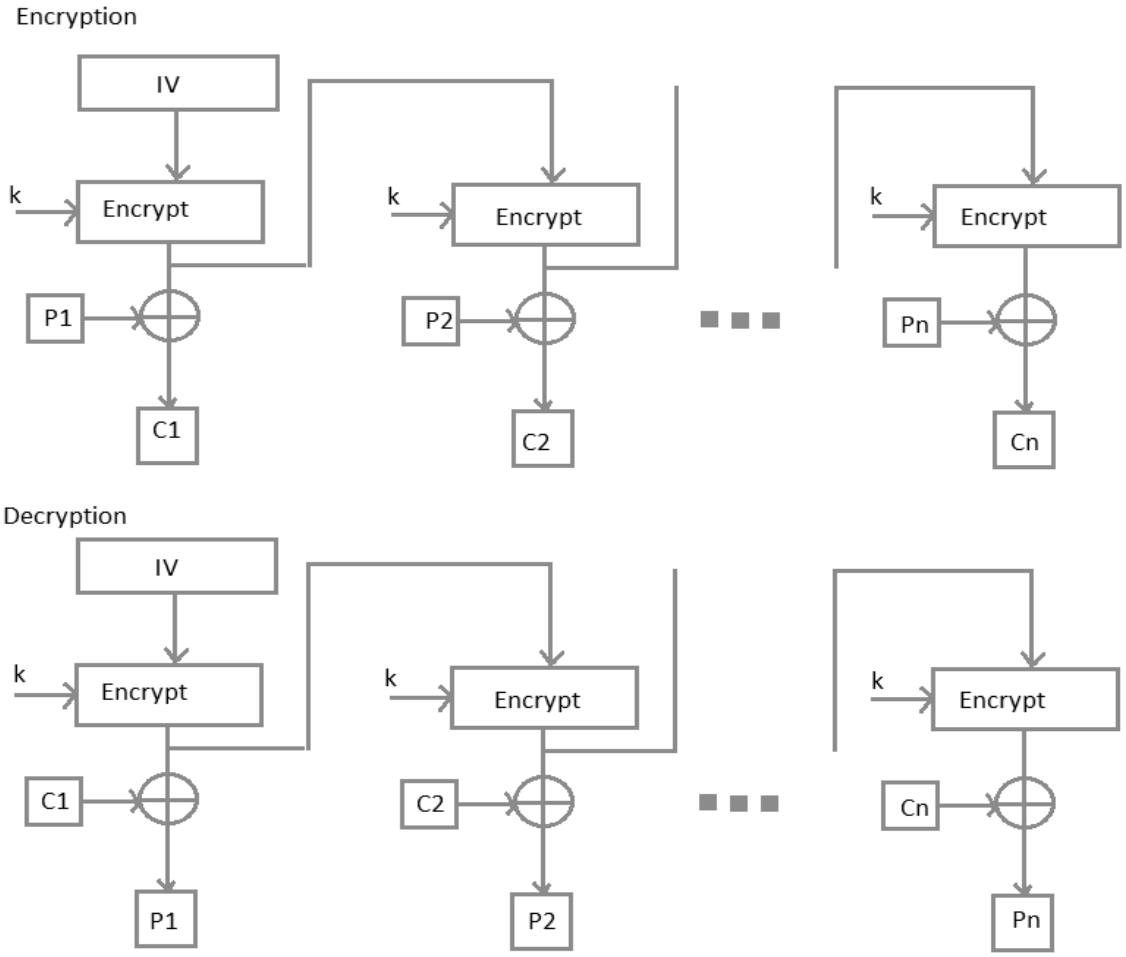


Advantages of CFB –

- Since, there is some data loss due to the use of shift register, thus it is difficult for applying cryptanalysis.

Output Feedback Mode (OFB) –

The output feedback mode follows nearly the same process as the Cipher Feedback mode except that it sends the encrypted output as feedback instead of the actual cipher which is XOR output. In this output feedback mode, all bits of the block are sent instead of sending selected *s* bits. The Output Feedback mode of block cipher holds great resistance towards bit transmission errors. It also decreases the dependency or relationship of the cipher on the plaintext.



Advantages of OFB –

- In the case of CFB, a single bit error in a block is propagated to all subsequent blocks. This problem is solved by OFB as it is free from bit errors in the plaintext block.

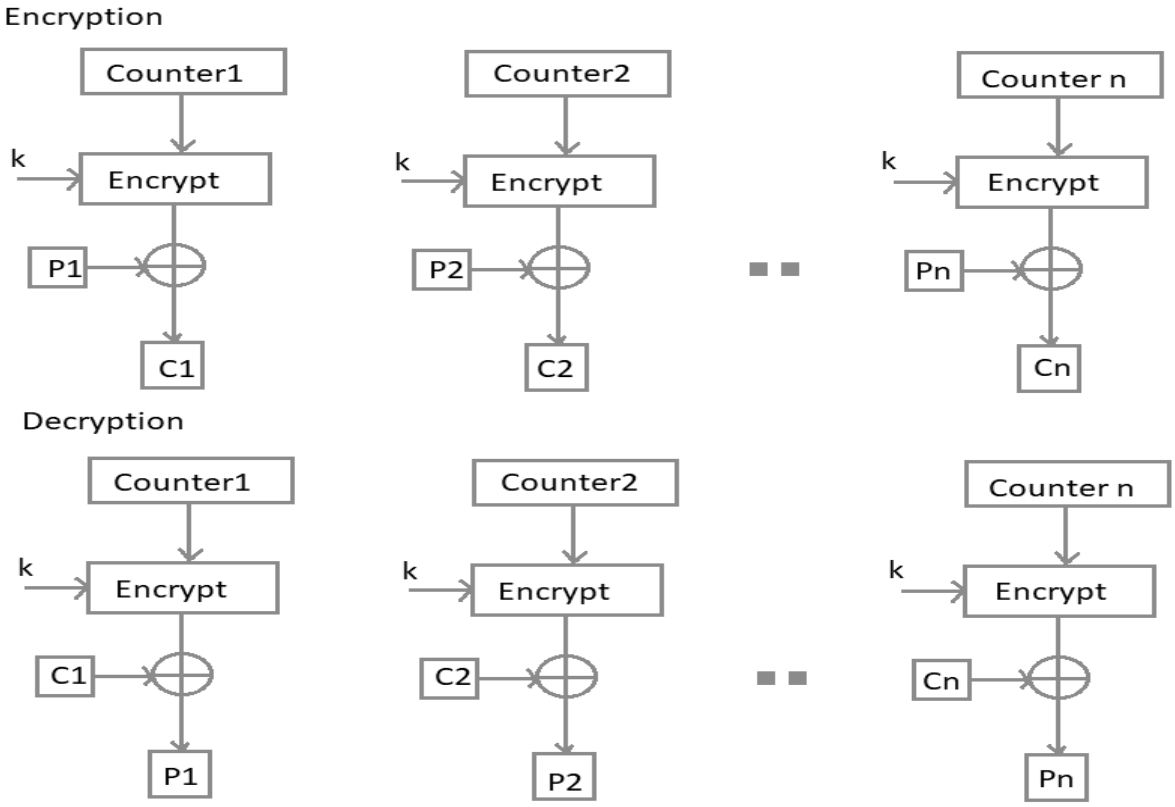
Counter Mode –

The Counter Mode or CTR is a simple counter-based block cipher implementation. Every time a counter-initiated value is encrypted and given as input to XOR with plaintext which results in cipher text block. The CTR mode is independent of feedback use and thus can be implemented in parallel.

Advantages of Counter –

- Since there is a different counter value for each block, the direct plaintext and cipher text relationship is avoided. This means that the same plain text can map to different cipher text.
- Parallel execution of encryption is possible as outputs from previous stages are not chained as in the case of CBC.

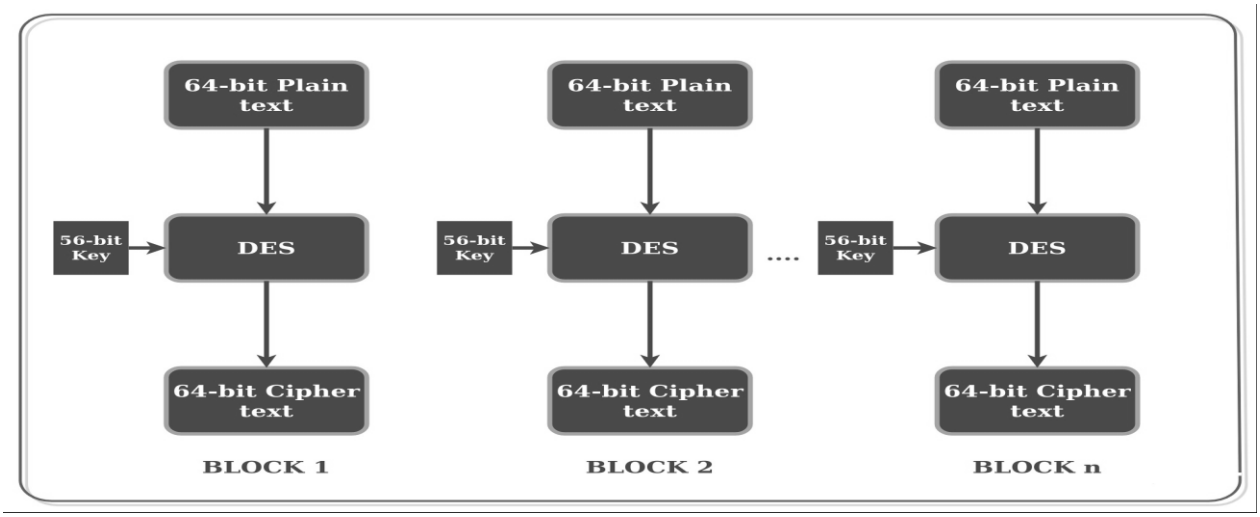
Its simple implementation is shown below:



BLOCK CIPHER ALGORITHMS

DATA ENCRYPTION STANDARD (DES ALGORIHM)

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 cipher text. 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.

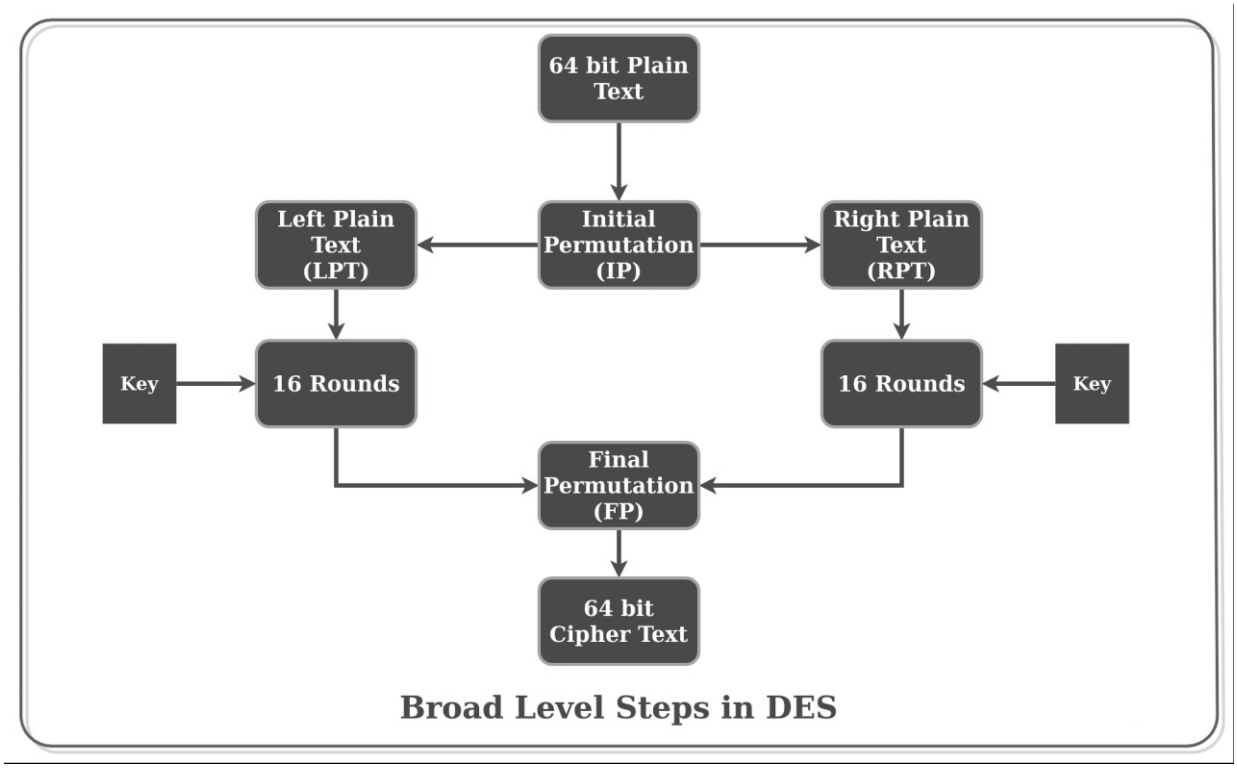
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

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 cipher text.



Broad Level Steps in DES

Initial Permutation (IP) –

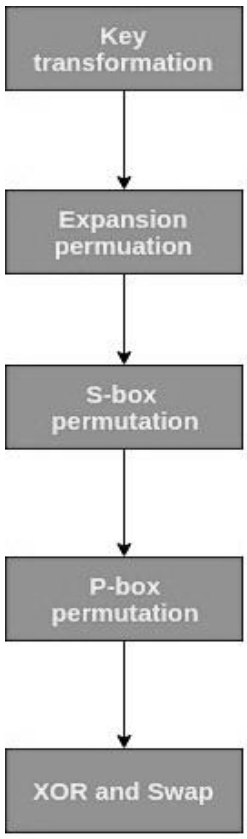
As we have noted, the initial permutation (IP) happens only once and it happens before the first round. It suggests how the transposition in IP should proceed, as shown in the figure. For example, it says that the IP replaces the first bit of the original plain text block with the 58th bit of the original 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.

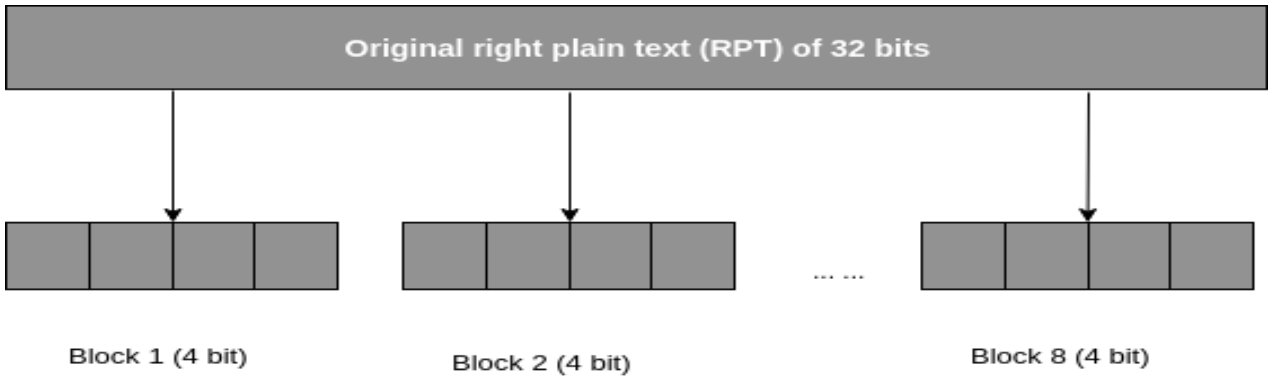
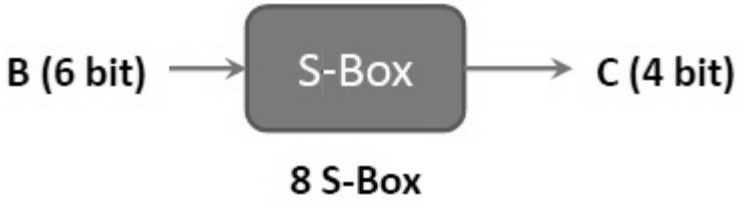


Figure - division of 32 bit RPT into 8 bit blocks

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.

Step 3: S-Box substitution:-

S-boxes are the only *non-linear* elements in DES design
Each of the unique selection functions S1, S2**,..., S8**, takes a 6-bit block as input and yields a 4-bit block as output.



• S = matrix 4x16, values from 0 to 15

• B (6 bit long) = b1b2b3b4b5b6
– b1b6
r = row of the matrix (2 bits: 0, 1, 2, 3)
– b2b3b4b5
-c = column of the matrix (4 bits: 0, 1…15)
• C (4 bit long) = Binary representation of S(r, c)

Example (S1)

Row #	S_1	1	2	3	…		7								15	Column #
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S(i, j) <16, can be represented with 4 bits

Example: B =101111

b_1b_6 = 11 = row 3

$b_2b_3b_4b_5$= 0111 = column 7

C=7=0111

Step 4: The P-Box Permutation: - The 32-bit output of the S-box substitution is permuted according to a P-box. This **permutation maps each input bit to an output position**; no bits are used twice and no bits are ignored. This is called a straight permutation or just a permutation.

Advanced Encryption Standard (AES)

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

- AES is a block cipher.
- The key size can be 128/192/256 bits.
- Encrypts data in blocks of 128 bits each.

That means it takes 128 bits as input and outputs 128 bits of encrypted cipher text as output. AES relies on substitution-permutation network principle which means it is performed using a series of linked operations which involves replacing and shuffling of the input data.

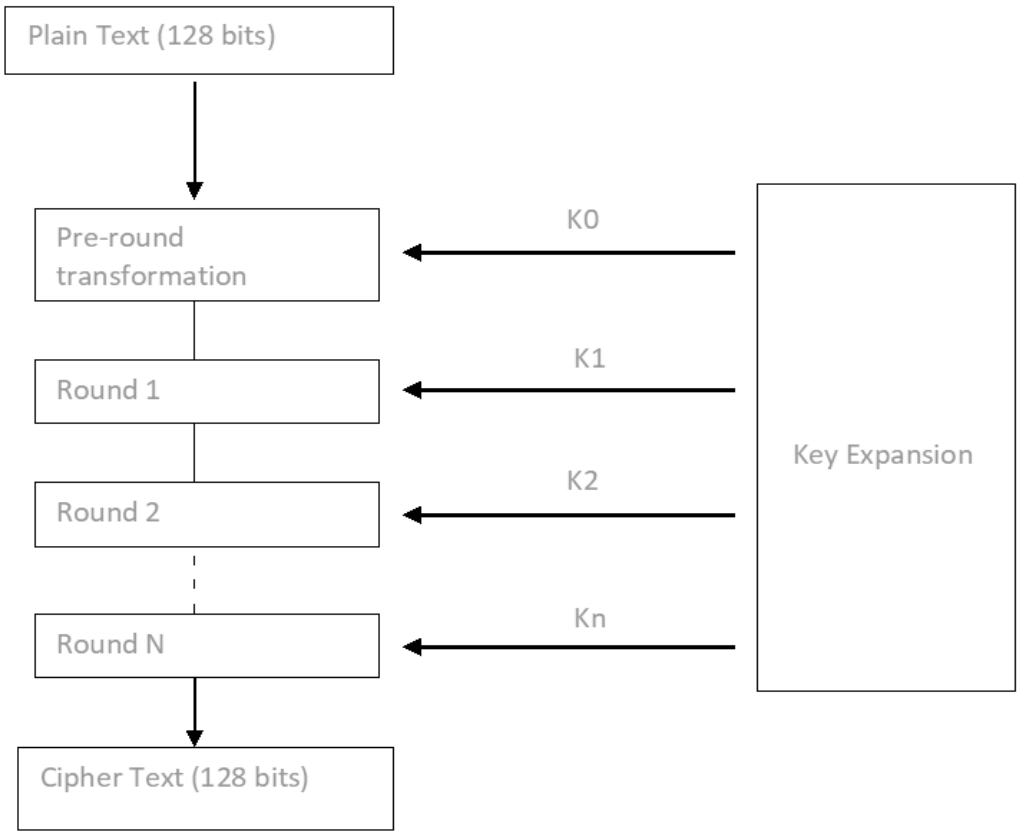
Working of the cipher:
AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

The number of rounds depends on the key length as follows :

- 128 bit key – 10 rounds
- 192 bit key – 12 rounds
- 256 bit key – 14 rounds

Creation of Round keys:

A Key Schedule algorithm is used to calculate all the round keys from the key. So the initial key is used to create many different round keys which will be used in the corresponding round of the encryption.



AES Encryption:

AES considers each block as a 16 byte (4 byte x 4 byte = 128) grid in a column major arrangement.

```
[ b0 | b4 | b8 | b12 |
| b1 | b5 | b9 | b13 |
| b2 | b6 | b10| b14 |
| b3 | b7 | b11| b15 ]
```

Each round comprises of 4 steps:

- SubBytes
- ShiftRows
- MixColumns
- Add Round Key

The last round doesn't have the MixColumns round.

The SubBytes does the substitution and ShiftRows and MixColumns performs the permutation in the algorithm.

SubBytes:
This step implements the substitution.
In this step each byte is substituted by another byte. It's performed using a lookup table also called the S-box. This substitution is done in a way that a byte is never substituted by itself and also not substituted by another byte which is a compliment of the current byte. The result of this step is a 16 byte (4 x 4) matrix like before.

The next two steps implement the permutation.

ShiftRows:
This step is just as it sounds. Each row is shifted a particular number of times.

- The first row is not shifted
- The second row is shifted once to the left.
- The third row is shifted twice to the left.
- The fourth row is shifted thrice to the left.

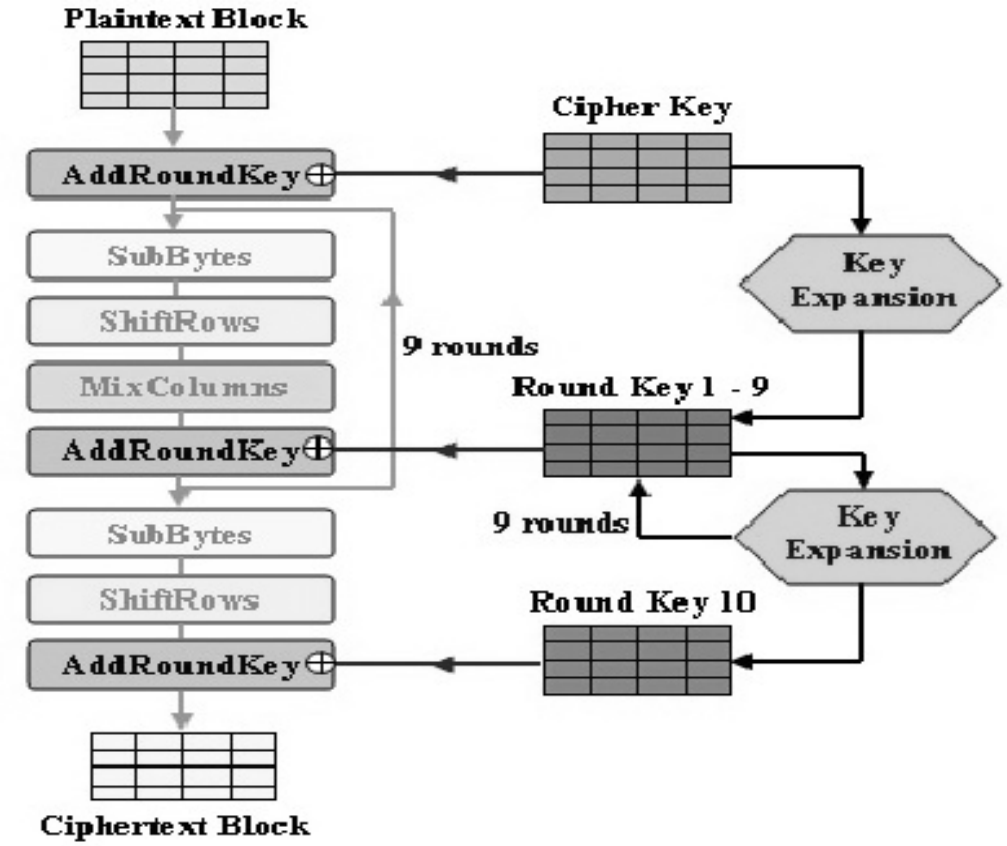
(A left circular shift is performed.)

```
[ b0  | b1  | b2  | b3  ]          [ b0  | b1  | b2  | b3  ]
| b4  | b5  | b6  | b7  |    ->    | b5  | b6  | b7  | b4  |
| b8  | b9  | b10 | b11 |          | b10 | b11 | b8  | b9  |
[ b12 | b13 | b14 | b15 ]          [ b15 | b12 | b13 | b14 ]
```

MixColumns:
This step is basically a matrix multiplication. Each column is multiplied with a specific matrix and thus the position of each byte in the column is changed as a result.
This step is skipped in the last round.

```
[ c0 ]          [ 2  3  1  1 ]  [ b0 ]
| c1 |  =       | 1  2  3  1 |     | b1 |
| c2 |       | 1  1  2  3 |     | b2 |
[ c3 ]      [ 3  1  1  2 ]     [ b3 ]
```

Add Round Keys:
Now the resultant output of the previous stage is XOR-ed with the corresponding round key. Here, the 16 bytes is not considered as a grid but just as 128 bits of data.



After all these rounds 128 bits of encrypted data is given back as output. This process is repeated until all the data to be encrypted undergoes this process.

AES Decryption:

The stages in the rounds can be easily undone as these stages have an opposite to it which when performed reverts the changes. Each 128 blocks goes through the 10,12 or 14 rounds depending on the key size.

The stages of each round in decryption are as follows:

- Add round key
- Inverse MixColumns
- ShiftRows
- Inverse SubByte

The decryption process is the encryption process done in reverse so i will explain the steps with notable differences.

Inverse MixColumns:

This step is similar to the MixColumns step in encryption, but differs in the matrix used to carry out the operation.

$$\begin{bmatrix} b0 \\ b1 \\ b2 \\ b3 \end{bmatrix} = \begin{bmatrix} 14 & 11 & 13 & 9 \\ 9 & 14 & 11 & 13 \\ 13 & 9 & 14 & 11 \\ 11 & 13 & 9 & 14 \end{bmatrix} \begin{bmatrix} c0 \\ c1 \\ c2 \\ c3 \end{bmatrix}$$

Inverse SubBytes:

Inverse S-box is used as a lookup table and using which the bytes are substituted during decryption.

AES instruction set is now integrated into the CPU (offers throughput of several GB/s) to improve the speed and security of applications that use AES for encryption and decryption. Even though it's been 20 years since its introduction we have failed to break the AES algorithm as it is infeasible even with the current technology. Till date the only vulnerability remains in the implementation of the algorithm.

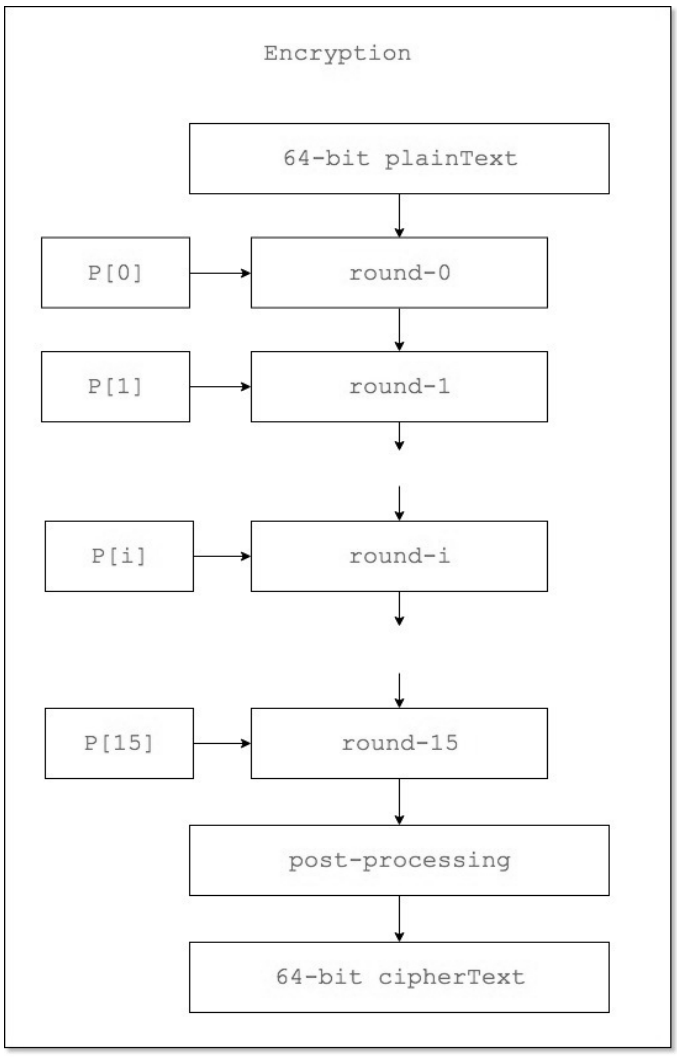
Blowfish Algorithm

Blowfish is an encryption technique designed by **Bruce Schneier** in 1993 as an alternative to DES Encryption Technique. It is significantly faster than DES and provides a good encryption rate with no effective cryptanalysis technique found to date. It is one of the first, secure block ciphers not subject to any patents and hence freely available for anyone to use.

1. **blockSize**: 64-bits
2. **keySize**: 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:



Step1: Generation of subkeys:

- 18 subkeys {P[0]…P[17]} are needed in both encryption as well as decryption process and the same 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 sub key 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.

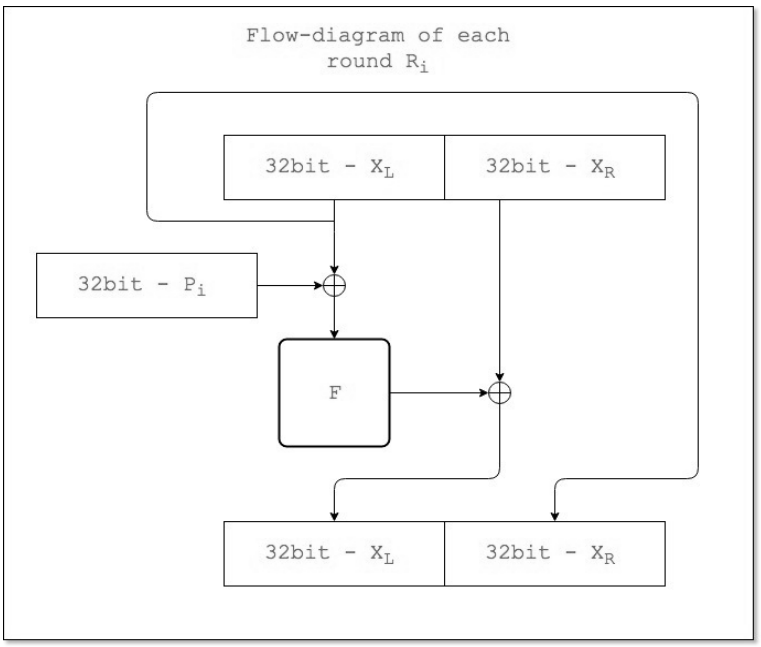
Step2: initialize Substitution Boxes:

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

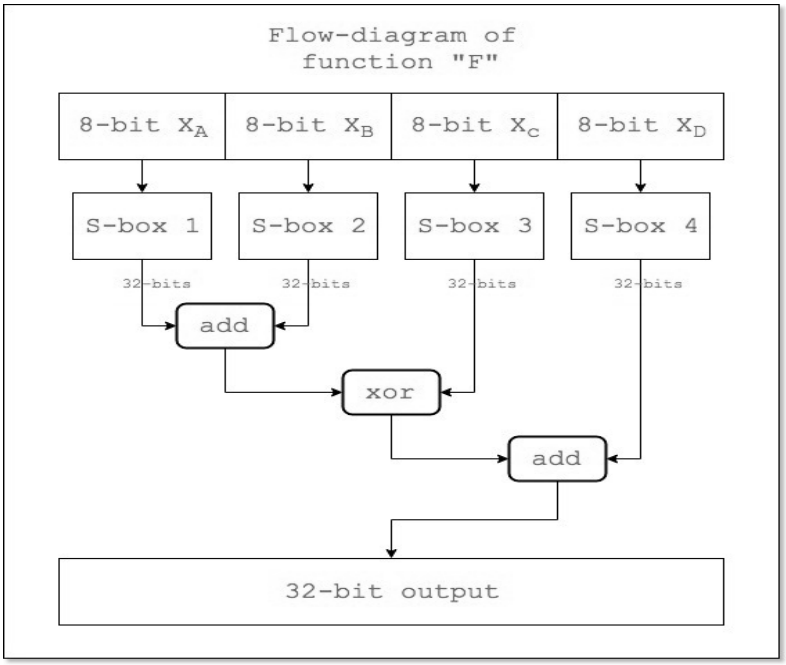
Step3: Encryption:

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.) from previous round and corresponding subkey(Pi). The description of each round is as follows:

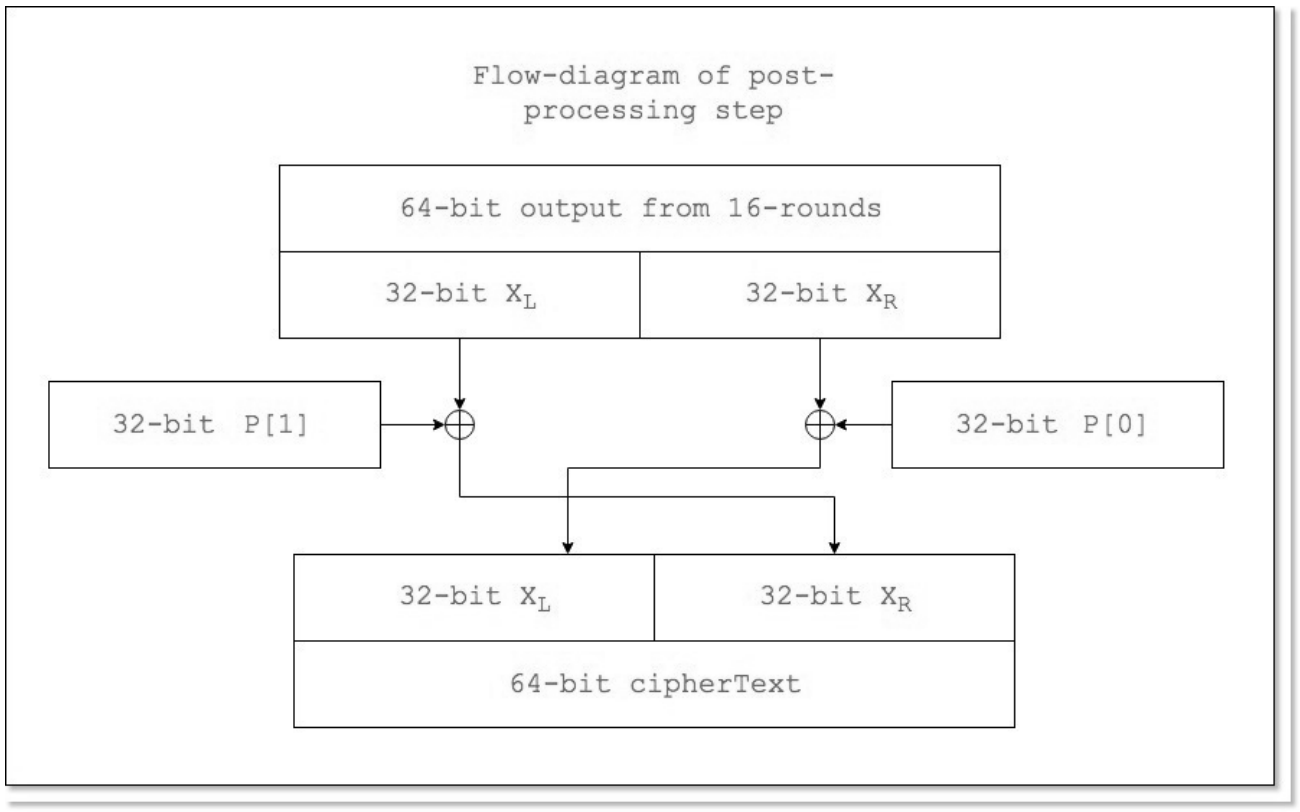


The description of the function" F" is as follows:



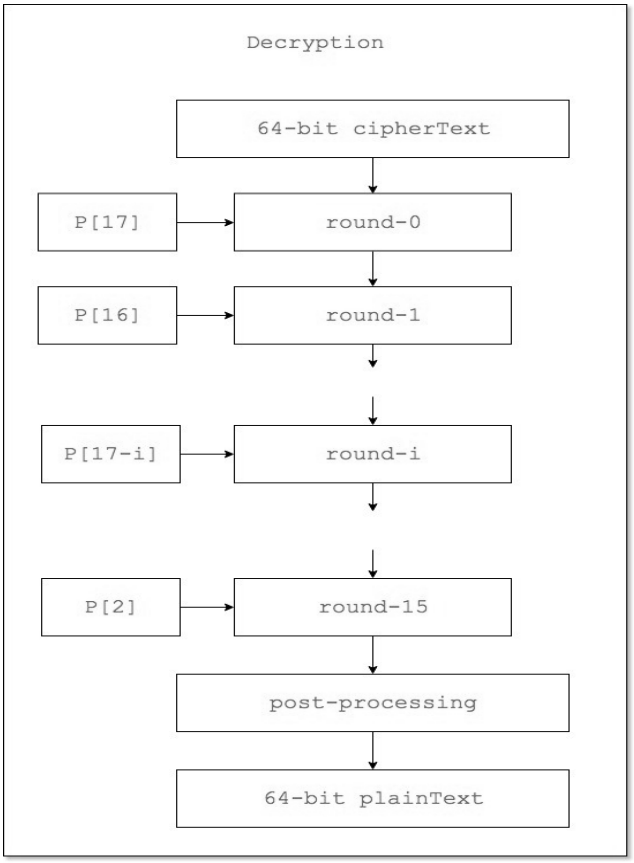
Here the function "add" is addition modulo 2^{32} .

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



Blowfish Decryption

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



DIFFERENTIAL AND LINEAR CRYPTANALYSIS

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

Differential Cryptanalysis

- One of the most significant advances in cryptanalysis in recent years is differential cryptanalysis.
- Differential cryptanalysis was not reported in the open literature until 1990.
- The first published effort appears to have been the cryptanalysis of a block cipher called FEAL by Murphy. This was followed by a number of papers by Biham and Shamir, who demonstrated this form of attack on a variety of encryption algorithms and hash functions.
- The most publicized results for this approach have been those that have application to DES. Differential cryptanalysis is the first published attack that is capable of breaking DES in less than 255 complexity.
- The scheme, as reported successfully crypt analyze DES with an effort on the order of 247 encryptions, requiring 247 chosen plaintexts. Although 247 is certainly significantly less than 255 the need for the adversary to find 247 chosen plaintexts makes this attack of only theoretical interest.

Differential Cryptanalysis Attack

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

We begin with a change in notation for DES. Consider the original plaintext block m to consist of two halves m0, m1. Each round of DES maps the right-hand input into the left-hand output and sets the right-hand output to be a function of the left-hand input and the sub key for this round. So, at each round, only one new 32-bit block is created. If we label each new block mi (2 … i … 17), then the intermediate message halves are related as follows:

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

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

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

Now, suppose that many pairs of inputs to f with the same difference yield the same output difference if the same sub key is used. To put this more precisely, let us say that *X may cause Y with probability p*, if for a fraction *p* of the pairs in which the input XOR is *X*, the output XOR equals *Y*. We want to suppose that there are a number of values of *X* that have high probability of causing a particular output difference.

Therefore, if we know Δm_{i-1} and Δm_i with high probability, then we know Δm_{i+1} with high probability. Furthermore, if a number of such differences are determined, it is feasible to determine the sub key used in the function f.

The overall strategy of differential cryptanalysis is based on these considerations for a single round. The procedure is to begin with two plaintext messages *m* and *m*' with a given difference and trace through a probable pattern of differences after each round to yield a probable difference for the cipher text.

Actually, there are two probable patterns of differences for the two 32-bit halves: $(\Delta m_{17} || m_{16})$. Next, we submit *m* and *m*' for encryption to determine the actual difference under the unknown key and compare the result to the probable difference. If there is a match,

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

then we suspect that all the probable patterns at all the intermediate rounds are correct. With that assumption, we can make some deductions about the key bits. This procedure must be repeated many times to determine all the key bits.

The probabilities shown on the right refer to the probability that a given set of intermediate differences will appear as a function of the input differences. Overall, after three rounds, the probability that the output difference is as shown is equal to
0.25 * 1 * 0.25 = 0.0625.

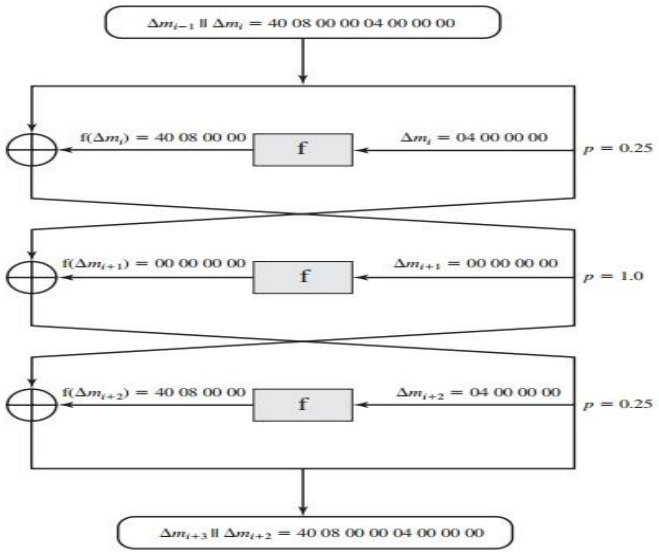


Figure 3.8 Differential Propagation through Three Rounds of DES (numbers in hexadecimal)

Linear Cryptanalysis

This attack is based on finding linear approximations to describe the transformations performed in DES. This method can find a DES key given 243 known plaintexts, as compared to 247 chosen plaintexts for differential cryptanalysis. Although this is a minor improvement, because it may be easier to acquire known plaintext rather than chosen plaintext, it still leaves linear cryptanalysis infeasible as an attack on DES. So far, little work has been done by other groups to validate the linear cryptanalytic approach.

We now give a brief summary of the principle on which linear cryptanalysis is based. For a cipher with n-bit plaintext and cipher text blocks and an m-bit key, let the plaintext block be labeled $P[1], \ldots P[n]$, the cipher text block $C[1], \ldots C[n]$, and the key $K[1], \ldots, K[m]$. Then define

$$A[i, j, \ldots, k] = A[i] \oplus A[j] \oplus \ldots \oplus A[k]$$

The objective of linear cryptanalysis is to find an effective *linear* equation of the form:

$$P[\alpha_1, \alpha_2, \ldots, \alpha_a] \oplus C[\beta_1, \beta_2, \ldots, \beta_b] = K[\gamma_1, \gamma_2, \ldots, \gamma_c]$$

(Where x = 0 or 1; 1 <= a; b <= n; c <= m; and where the a, b, and g terms represent fixed, unique bit locations) that holds with probability p != 0.5. The further p is from 0.5, the more effective the equation. Once a proposed relation is determined, the pro- cedure is to compute the results of the left-hand side of the preceding equation for a large number of plaintext–ciphertext pairs. If the result is 0 more than half the time, assume $K[\gamma_1, \gamma_2, \ldots, \gamma_c] = 0$. If it is 1 most of the time, assume $K[\gamma_1, \gamma_2, \ldots, \gamma_c] = 1$. This gives us a linear equation on the key bits. Try to get more such relations so that we can solve for the key bits. Because we are dealing with linear equations, the problem can be approached one round of the cipher at a time, with the results combined.

Stream cipher algorithm

RC4 Encryption Algorithm

RC4 is a stream cipher and variable-length key algorithm. This algorithm encrypts one byte at a time (or larger units at a time).
A key input is pseudorandom bit generator that produces a stream 8-bit number that is unpredictable without knowledge of input key, The output of the generator is called key-stream, is combined one byte at a time with the plaintext stream cipher using X-OR operation.

Example:

RC4 Encryption

10011000 XOR 01010000 = 11001000

RC4 Decryption

11001000 XOR 01010000 = 10011000

Key-Generation Algorithm –
A variable-length key from 1 to 256 bytes is used to initialize a 256-byte state vector S, with elements S[0] to S[255]. For encryption and decryption, a byte k is generated from S by selecting one of the 255 entries in a systematic fashion, then the entries in S are permuted again.

1. Key-Scheduling Algorithm(KSA):

Initialization: The entries of S are set equal to the values from 0 to 255 in ascending order, a temporary vector T, is created.
If the length of the key k is 256 bytes, then k is assigned to T. Otherwise, for a key with length(k-len) bytes, the first k-len elements of T as copied from K, and then K is repeated as many times as necessary to fill T. The idea is illustrated as follow:

```
for
i = 0 to 255 do S[i] = i;
T[i] = K[i mod k - len];
```

We use T to produce the initial permutation of S. Starting with S[0] to S[255], and for each S[i] algorithm swap it with another byte in S according to a scheme dictated by T[i], but S will still contain values from 0 to 255 :

```
j = 0;
for
    i = 0 to 255 do
    {
        j = (j + S[i] + T[i])mod 256;
        Swap(S[i], S[j]);
    }
```

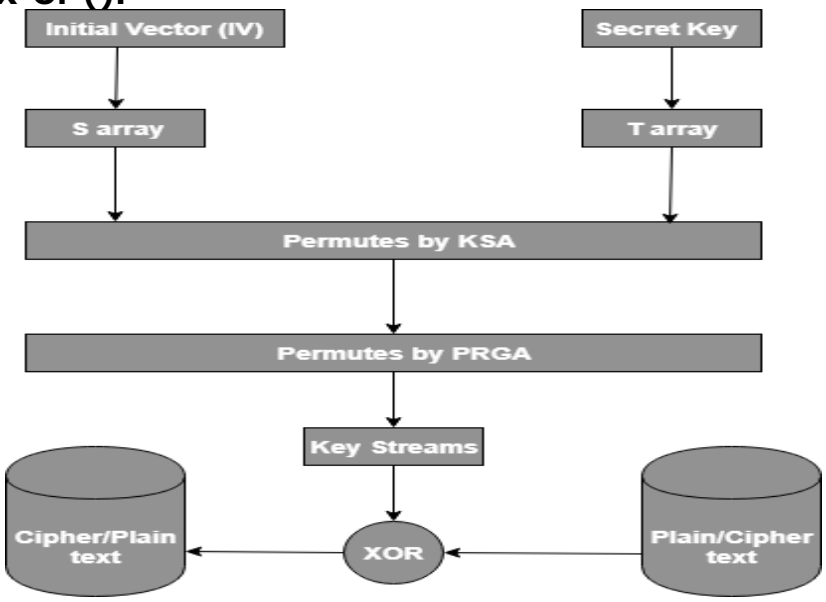
2. Pseudo random generation algorithm (PRGA):

Once the vector S is initialized, the input key will not be used. In this step, for each S[i] algorithm swap it with another byte in S according to a scheme dictated by the current configuration of S. After reaching S[255] the process continues, starting from S[0] again

```
i, j = 0;
while (true)
       i = (i + 1)mod 256;
j = (j + S[i])mod 256;
```

Swap(S[i], S[j]);
t = (S[i] + S[j])mod 256;
k = S[t];

3. Encrypt using X-Or ():

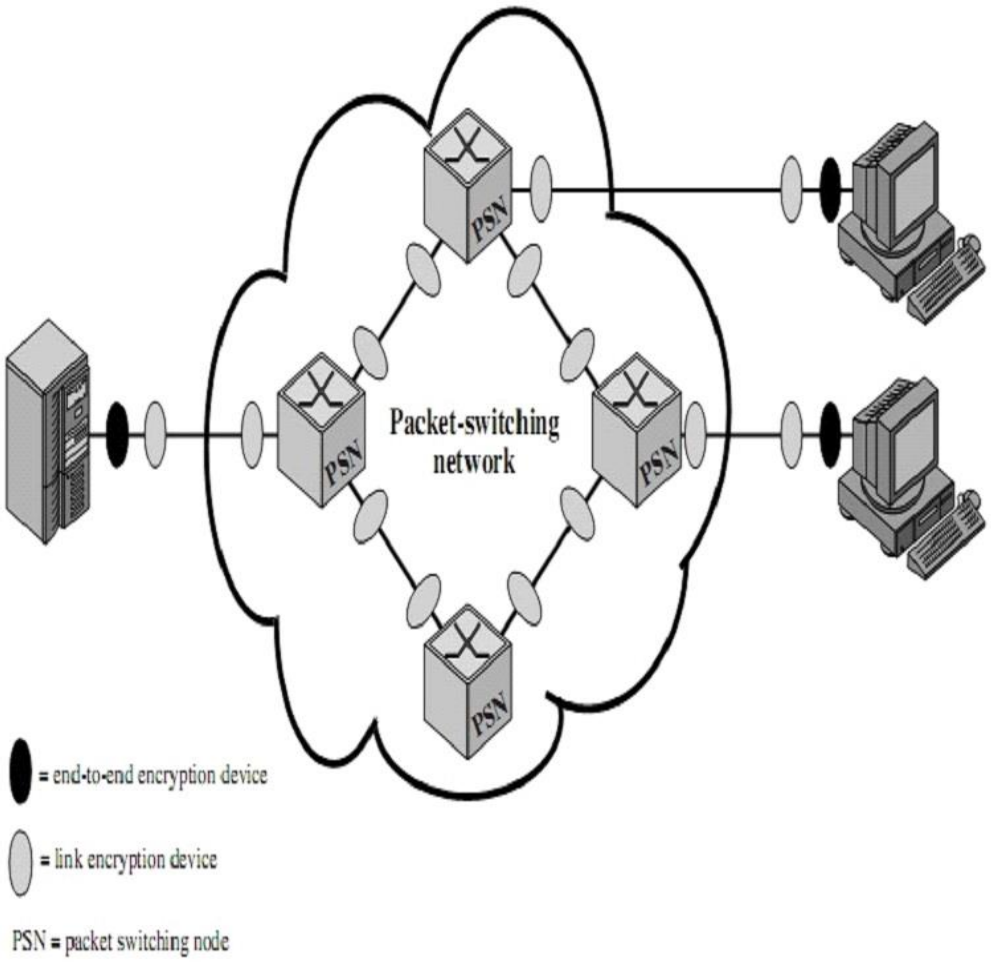


Placement of Encryption Function:-

If encryption is to be used to counter attacks on confidentiality, we need to decide what to encrypt and where the encryption function should be located. To begin, this section examines the potential locations of security attacks and then looks at the two major approachesto encryption placement: **link and end to end.**

There are a large number of locations at which an attack can occur. Furthermore, for wide area communications, many of these locations are not under the physical control of the end user. Even in the case of local area networks, in which physical security measures are possible, there is always the threat of the disgruntled employee.

The most powerful and most common approach to securing the points of vulnerability highlighted in the preceding section is encryption. If encryption is to be used to counter these attacks, then we need to decide what to encrypt and where the encryption gear should be located. There are two fundamental alternatives: **link encryption and end-to-end encryption.**



Encryption Across a Packet-Switching Network

Link to Link Encryption:

With link encryption, each vulnerable communications link is equipped on both ends with an encryption device. Thus, all traffic over all communications links is secured. One of its disadvantages is that the message must be decrypted each time it enters a switch because the switch must read the address (logical connection number) in the packet header in order to route the frame. Thus, the message is vulnerable at each switch. If working with a public network, the user has no control over the security of the nodes.

Several implications of link encryption should be noted. For this strategy to be effective, all the potential links in a path from source to destination must use link encryption. Each pair of nodes that share a link **should share a unique key, with a different key used oneach link.** Thus, many keys must be provided.

End-To-End Encryption

With end-to-end encryption, the encryption process is carried out at the two end systems. The source host or terminal encrypts the data. The data in encrypted form are

then transmitted unaltered across the network to the destination terminal or host. The destination shares a key with the source and so is able to decrypt the data. This plan seems to secure the transmission against attacks on the network links or switches. Thus, end-to-end encryption relieves the end user of concerns about the degree of security of networks and links that support the communication. There is, however, still a weak spot.

Consider the following situation. A host connects to a frame relay or ATM network, sets up a logical connection to another host, and is prepared to transfer data to that other host by using end-to-end encryption. Data are transmitted over such a network in the form of packets that consist of a header and some user data. The frame relay or ATM switch will receive an encrypted packet and be unable to read the header. Therefore, it will not be able to route the packet. It follows that the host may encrypt only the user data portion of the packet and must leave the header in the clear.

Thus, with end-to-end encryption, the user data are secure. However, the traffic pattern is not, because packet headers are transmitted in the clear. On the other hand, end-to-end encryption does provide a degree of authentication. If two end systems share an encryption key, then a recipient is assured that any message that it receives comes from the alleged sender, because only that sender shares the relevant key. Such authentication is not inherent in a link encryption scheme.

Key Distribution

For symmetric encryption to work, the two parties to an exchange must share the same key, and that key must be protected from access by others. Furthermore, frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key. Therefore, the term that refers to the means of delivering a key to two parties who wish to exchange data, without allowing others to see the key. For two parties A and B, key distribution can be achieved in a number of ways, as follows:

1. A can select a key and physically deliver it to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

Physical delivery (1 & 2) is simplest - but only applicable when there is personal contact between recipient and key issuer. This is fine for link encryption where devices & keys occur in pairs, but does not scale as number of parties who wish to communicate grows. 3 is mostly based on 1 or 2 occurring first.

A third party, whom all parties trust, can be used as a **trusted intermediary** to mediate the establishment of secure communications between them (4). Must trust intermediary not to abuse the knowledge of all session keys. As number of parties grow, some variant of 4 is only practical solution to the huge growth in number of keys potentially needed.

Key distribution centre: (KDC)

- The use of a key distribution center is based on the use of a hierarchy of keys. At aminimum, two levels of keys are used.
- Communication between end systems is encrypted using a temporary key, often referred toas a **session key**.
- Typically, the session key is used for the duration of a logical connection and then discarded
- **Master key** is shared by the key distribution center and an end system or user and used toencrypt the session key.

Key Distribution Scenario:

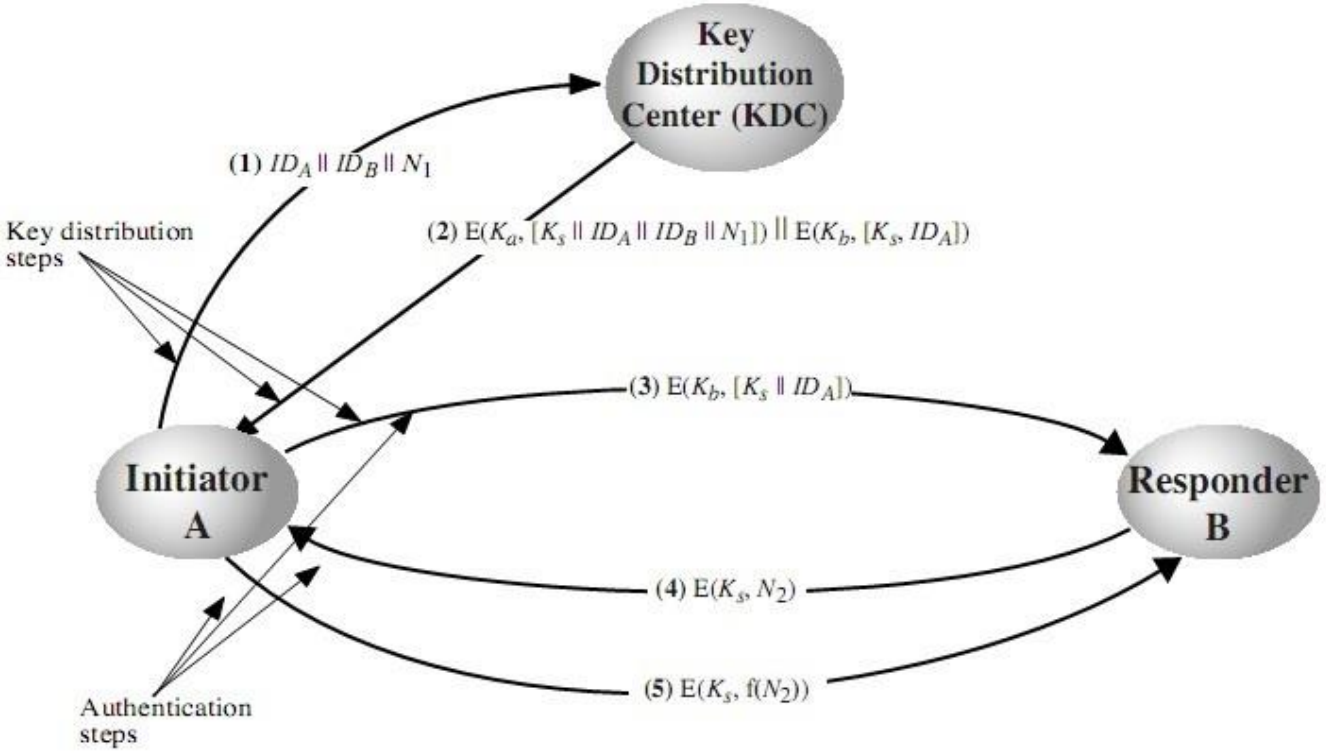


Figure 7.9 Key Distribution Scenario

Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection. A has a master key, K_a, known only to itself and the KDC; similarly, B shares the master key K_b with the KDC. The following steps occur:

1. A issues a request to the KDC for a session key to protect a logical connection to B. The message includes the identity of A and B and a unique identifier, N_1, for this transaction, which we refer to as a **nonce**. The nonce may be a timestamp, a counter, or a random number; the minimum requirement is that it differs with each request. Also, to prevent masquerade, it should be difficult for an opponent to guess the nonce. Thus, a random number is a good choice for a nonce.
2. The KDC responds with a message encrypted using Ka Thus, A is the only one who can successfully read the message, and A knows that it originated at the KDC. The message includes two items intended for A:
    - The one-time session key, Ks, to be used for the session
    - The original request message, including the nonce, to enable A to match this response with the appropriate request

Thus, A can verify that its original request was not altered before reception by the KDC and, because of the nonce, that this is not a replay of some previous request.
In addition, the message includes two items intended for B:

- The one-time session key, Ks to be used for the session
- An identifier of A (e.g., its network address), IDA

These last two items are encrypted with Kb (the master key that the KDC shares with B).They are to be sent to B to establish the connection and prove A's identity.

3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely, $E(K_b, [K_s \| ID_A])$. Because this information is encrypted with K_b, it is protected from eavesdropping. B now knows the session key (K_s), knows that the other party is A (from ID_A), and knows that the information originated at the KDC (because it is encrypted using K_b).

At this point, a session key has been securely delivered to A and B, and they may begintheir protected exchange. However, two additional steps are desirable:

4. Using the newly minted session key for encryption, B sends a nonce, N_2, to A.
5. Also using K_s, A responds with $f(N_2)$, where f is a function that performs some transformation on N_2 (e.g., adding one).

These steps assure B that the original message it received (step 3) was not a replay.

Note that the actual key distribution involves only steps 1 through 3 but that steps 4 and 5, aswell as 3, perform an authentication function.

Major Issues with KDC:

For very large networks, a hierarchy of KDCs can be established. For communication among entities within the same local domain, the local KDC is responsible for key distribution. If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a (hierarchy of) global KDC(s)

To balance security & effort, a new session key should be used for each new connection-oriented session. For a connectionless protocol, a new session key is used for a certain fixed period only or for a certain number of transactions.

An automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other, provided they trust the system to act on theirbehalf.

The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion. This requirement can be avoided if key distribution is fully decentralized.

In addition to separating master keys from session keys, may wish to define different types of session keys on the basis of use.

UNIT-3

MESSAGE AUTHENTICATION ALGORITHMS

AND HASH FUNCTIONS

1. MESSAGE AUTHENTICATION REQUIREMENTS

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

1. Disclosure: Release of message contents to any person or process not possessing the appropriate cryptographic key.

2. Traffic analysis: Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.

3. Masquerade: Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or non-receipt by someone other than the message recipient.

4. Content modification: Changes to the contents of a message, including insertion, deletion, transposition, and modification.

5. Sequence modification: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.

6. Timing modification: Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., data- gram) could be delayed or replayed.

7. Source repudiation: Denial of transmission of message by source.

8. Destination repudiation: Denial of receipt of message by destination.

Measures to deal with the first two attacks are in the realm of message confidentiality and are dealt with in Part One. Measures to deal with items (3) through (2) in the foregoing list are generally regarded as message authentication. Mechanisms for dealing specifically with item (7) come under the heading of digital signatures. Generally, a digital signature technique will also counter some or all of the attacks listed under items (3) through (6). Dealing with item (8) may require a combination of the use of digital signatures and a protocol designed to counter this attack.

In summary, message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness. A digital signature is an authentication technique that also includes measures to counter repudiation by the source.

2. MESSAGE AUTHENTICATION FUNCTIONS

Any message authentication or digital signature mechanism has two levels of functionality. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

This section 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, which serves as the authenticator
- **Message encryption**: The cipher text of the entire message serves as its authenticator
- **Message authentication code (MAC):** A function of the message and a secret key that produces a fixed-length value that serves as the authenticator.

Message Encryption:-

Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.

SYMMETRIC ENCRYPTION

Consider the straightforward use of symmetric encryption (fig a).

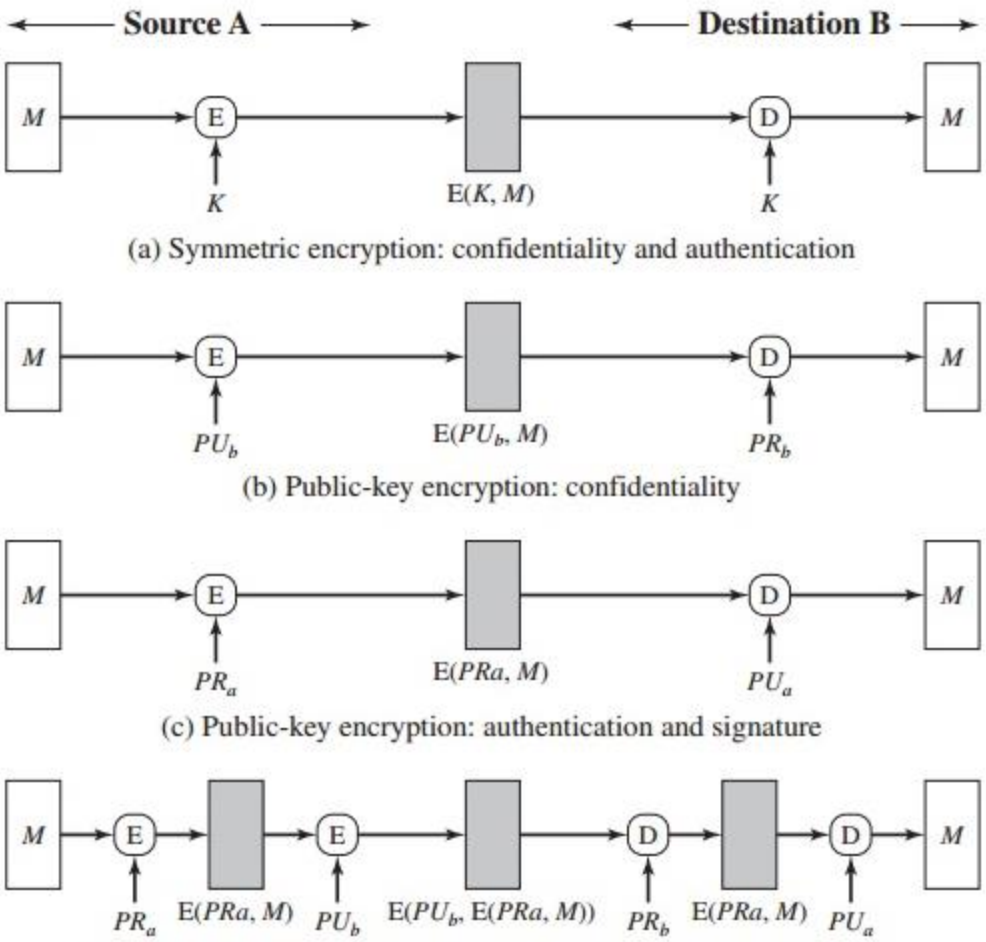
A message M transmitted from source A to destination B is encrypted using a secret key K shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.

In addition, B is assured that the message was generated by A. The message must have come from A, because A is the only other party that possesses K and therefore the only other party with the information necessary to construct cipher- text that can be decrypted with K. Furthermore, if M is recovered, B knows that none of the bits of M have been altered, because an opponent that does not know K would not know how to alter bits in the cipher text to produce the desired changes in the plaintext.

So we may say that symmetric encryption provides authentication as well as confidentiality. However, this flat statement needs to be qualified. Consider exactly what is happening at B. Given a decryption function D and a secret key K, the destination will accept any input X and produce output Y = D(K, X). If X is the cipher text of a legitimate message M produced by the corresponding encryption function, then Y is some plaintext message M. Otherwise, Y will likely be a meaningless sequence of bits. There may need to be some automated means of determining at B whether Y is legitimate plaintext and therefore must have come from A.

PUBLIC-KEY ENCRYPTION The straightforward use of public-key encryption (Figure b) provides confidentiality but not authentication. The source (A) uses the public key PU_B of the destination (B) to encrypt M. Because only B has the Corresponding private key PR_B, only B can decrypt the message. This scheme provides no authentication, because any opponent could also use B's public key to encrypt a message and claim to be A.

To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt (Figure c).



Basic Uses of Message Encryption

This provides authentication using the same type of reasoning as in the symmetric encryption case: The message must have come from A because A is the only party that possesses PR_A and therefore the only party with the information necessary to construct cipher text that can be decrypted with PU_A. Again, the same reasoning as before applies: There must be some internal structure to the plaintext so that the receiver can distinguish between well-formed plaintext and random bits.

Assuming there is such structure, then the scheme of Figure c does provide authentication. It also provides what is known as digital signature.1 Only A could have constructed the cipher text because only A possesses PR_A. Not even B, the recipient, could have constructed the cipher text. Therefore, if B is in possession of the cipher text, B has the means to prove that the message must have come from A. In effect, A has "signed" the message by using its private key to encrypt. Note that this scheme does not provide confidentiality. Anyone in possession of A's public key can decrypt the cipher text.

To provide both confidentiality and authentication, A can encrypt M first using its private key, which provides the digital signature, and then using B's public key, which provides confidentiality (Fig d).

The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

Message Authentication Code: (MAC)

An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a **cryptographic checksum or** MAC that is appended to the message. 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 = MAC (*K*, *M*)

Where

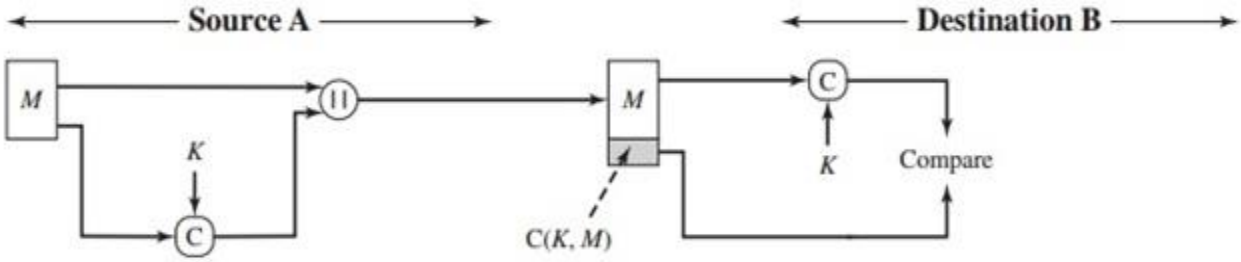
M = input message

C = MAC function

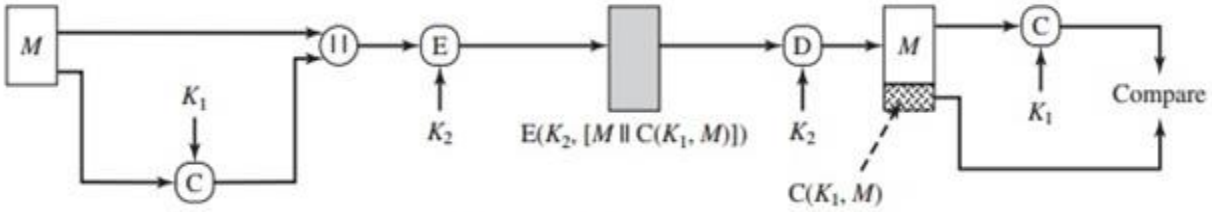
K = shared secret key

MAC = message authentication code

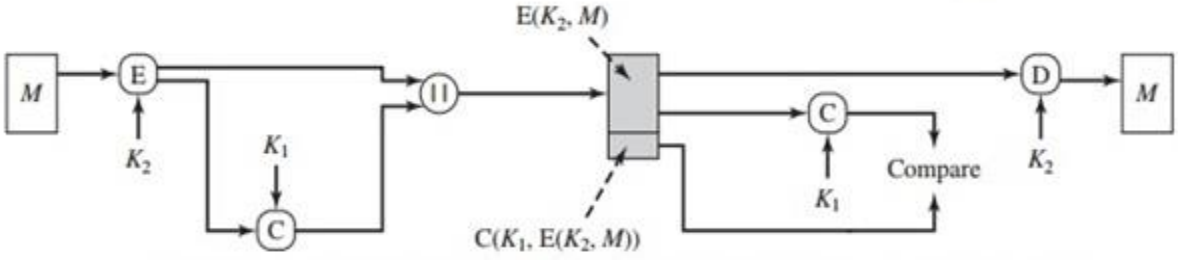
The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC (Figure a).



(a) Message authentication



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

Figure 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 received MAC matches the calculated MAC, then

1. 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.

2. 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.

3. If the message includes a sequence number (such as is used with HDLC, X.25, and 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. If an n-bit MAC is used, then there are 2n possible MACs, whereas there are N possible messages with N 77 2n. Furthermore, with a k-bit key, there are 2k possible keys.

For example, suppose that we are using 100-bit messages and a 10-bit MAC. Then, there are a total of 2100 different messages but only 210 different MACs. So, on average, each MAC value is generated by a total of 2100/210 = 290 different messages. If a 5-bit key is used, then there are 25 = 32 different mappings from the set of messages to the set of MAC values.

It turns out that, because of the mathematical properties of the authentication function, it is less vulnerable to being broken than encryption.

The process depicted in Figure a provides authentication but not confidentiality, because the message as a whole is transmitted in the clear. Confidentiality can be provided by performing message encryption either after (Figure b) or before (Figure c) the MAC algorithm. In both these cases, two separate keys are needed, each of which is shared by the sender and the receiver.

In the first case, the MAC is calculated with the message as input and is then concatenated to the message. The entire block is then encrypted.

In the second case, the message is encrypted first. Then the MAC is calculated using the resulting cipher text and is concatenated to the cipher text to form the transmitted block. Typically, it is preferable to tie the authentication directly to the plaintext, so the method of Figure b is used.

Hash Function

A hash value *h* is generated by a function H of the form

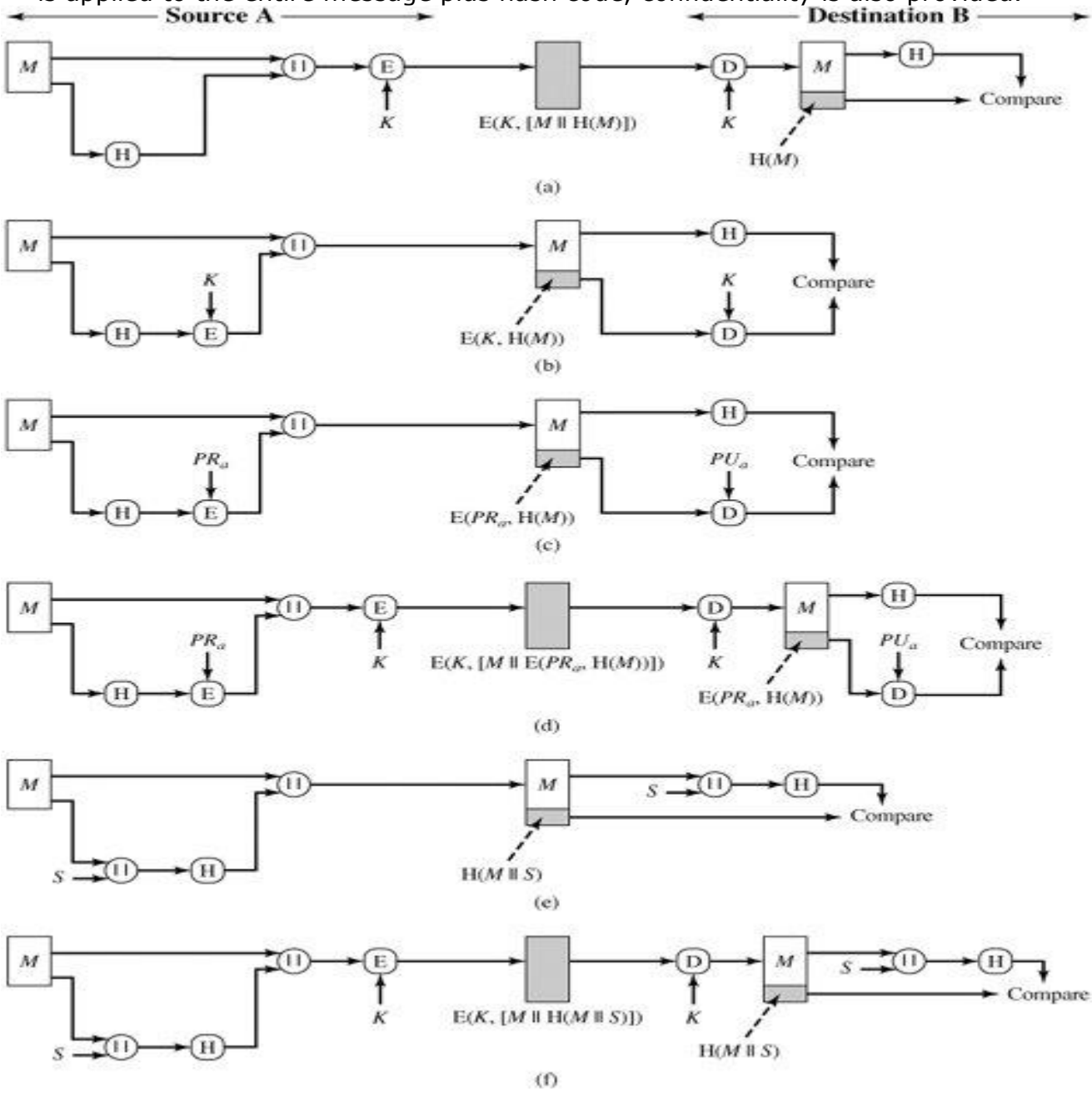
h = H(*M*) where *M* is a variable-length message and H(*M*) is the fixed-length hash value. The hash value is appended to the message at the source at a time when the message is assumed or known to be correct.

The receiver authenticates that message by recomputing the hash value. Because the hash function itself is not considered to be secret, some means is required to protect the hash value

A variation on the message authentication code is the one-way hash function. As with the message authentication code, a hash function accepts a variable-size message *M* as input and produces a fixed size output, referred to as a **hash code** H (*M*). Unlike a MAC, a hash code does not use a key but is a function only of the input message.

The hash code is also referred to as a **message digest** or **hash value**. The hash code is a function of all the bits of the message and provides an error-detection capability: A change to any bit or bits in the message results in a change to the hash code.

A. The message plus concatenated hash code is encrypted using symmetric encryption. This is identical in structure to the internal error control strategy shown in Figure a. The same line of reasoning applies: 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. Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality. Note that

the combination of hashing and encryption results in an overall function that is, in fact, a MAC (Figure a). That is, E(*K*, H(*M*)) is a function of a variable-length message *M* and a secret key *K*, and it produces a fixed-size output that is secure against an opponent who does not know the secret key.

C. Only the hash code is encrypted, using public-key encryption and using the sender's private key. As with (b), this provides authentication. It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.
D. If confidentiality as well as a digital signature is desired, then the message plus the private-key encrypted hash code can be encrypted using a symmetric secret key. This is a common technique.
E. 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.
F. Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code.

When confidentiality is not required, methods (b) and (c) have an advantage over those that encrypt the entire message in that less computation is required. Nevertheless, there has been growing interest in techniques that avoid encryption (Figure e).

Requirements for a Hash Function

The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. To be useful for message authentication, a hash function H must have the following properties.

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the one-way property.
5. for any given block x, it is computationally infeasible to find y x such that H(y) = H(x). This is sometimes referred to as weak collision resistance.
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as strong collision resistance.

Unfortunately, these terms are not used consistently. Alternate terms used in the literature include

- one-way hash function: (properties 4 and 5);
- Collision resistant hash function: (properties 4, 5, and 6);
- Weak one-way hash function :(properties 4 and 5);
- Strong one-way hash function: (properties 4, 5, and 6).

The first three properties are requirements for the practical application of a hash function to message authentication.

The fourth property, the one-way property, states that it is easy to generate a code given a message but virtually impossible to generate a message given a code. This property is important if the authentication technique involves the use of a secret value (Figure e). The secret value itself is not sent; however, if the hash function is not one way,

an attacker can easily discover the secret value: If the attacker can observe or intercept a transmission, the attacker obtains the message M and the hash code C = H(SAB||M). The attacker then inverts the hash function to obtain SAB||M = H1(C). Because the attacker now has both M and SAB||M, it is a trivial matter to recover SAB.

The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found. This prevents forgery when an encrypted hash code is used (Figures b and c). For these cases, the opponent can read the message and therefore generate its hash code.

However, because the opponent does not have the secret key, the opponent should not be able to alter the message without detection. If this property were not true, an attacker would be capable of the following sequence: First, observe or intercept a message plus its encrypted hash code; second, generate an unencrypted hash code from the message; third, generate an alternate message with the same hash code.

The sixth property refers to how resistant the hash function is to a type of attack known as the birthday attack.

Secure Hash Algorithm (SHA 512)

- The Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993.
- A revised version was issued as FIPS 180-1 in 1995 and is generally referred to as SHA-1. The actual standards document is entitled Secure Hash Standard.
- SHA is based on the hash function MD4 and its design closely models MD4. SHA-1 is also specified in RFC 3174, which essentially duplicates the material in FIPS 180-1, but adds a C code implementation.
- SHA-1 produces a hash value of 160 bits. In 2002, NIST produced a revised version of the standard, FIPS 180-2, that defined three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512 (Table).

Table Comparison of SHA Parameters

	SHA-1	SHA-256	SHA-384	SHA-512
Message digest size	160	256	384	512
Message size	$<2^{64}$	$<2^{64}$	$<2^{128}$	$<2^{128}$
Block size	512	512	1024	1024
Word size	32	32	64	64
Number of steps	80	64	80	80
Security	80	128	192	256

1. All sizes are measured in bits.
2. Security refers to the fact that a birthday attack on a message digest of size *n* produces a collision with a work factor of approximately $2^{n/2}$

SHA-512 Logic

The algorithm takes as input a message with a maximum length of less than 2128 bits and produces as output a 512-bit message digest. The input is processed in 1024-bit blocks. The following Figure depicts the overall processing of a message to produce a digest.

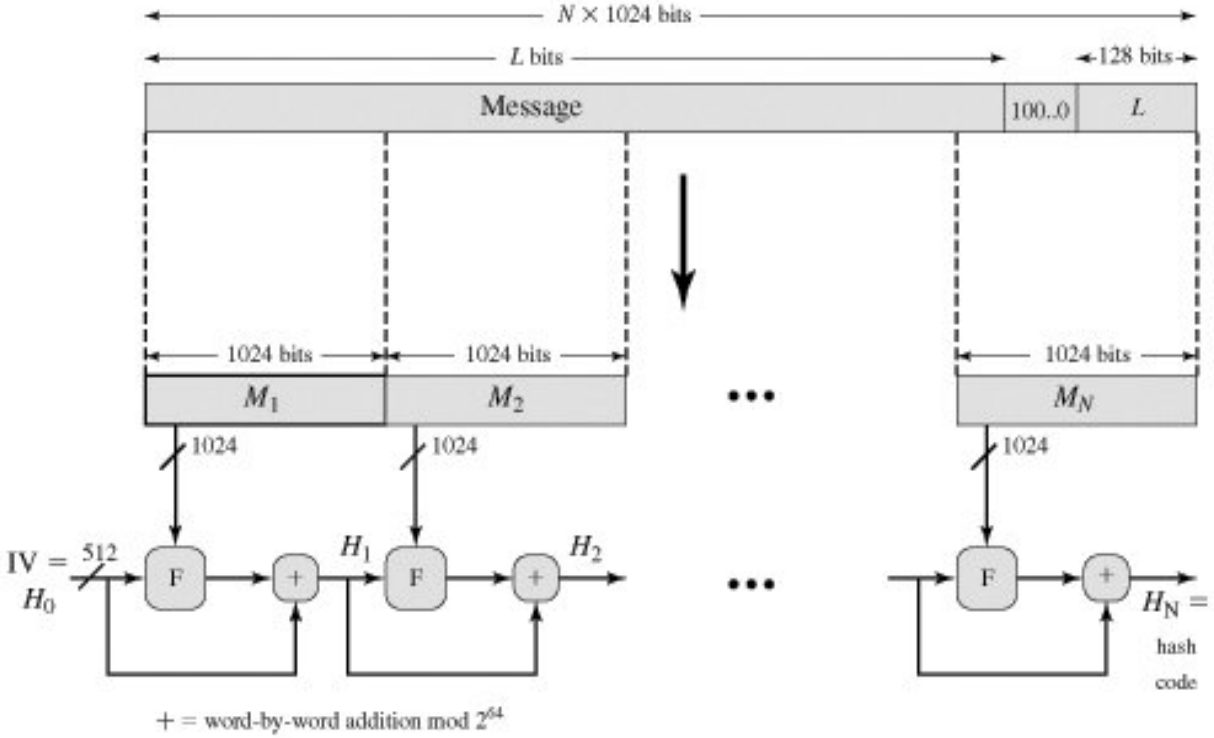


Figure 1: Message Digest Generation Using SHA-512

The processing consists of the following steps:

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

Step 2: Append length. A block of 128 bits is appended to the message. This block is treated as an unsigned 128-bit integer (most significant byte first) and contains the length of the original message (before the padding).

The outcome of the first two steps yields a message that is an integer multiple of 1024 bits in length. In Figure 12.1, the expanded message is represented as the sequence of 1024-bit blocks *M*1, *M*2,..., *M*N, so that the total length of the expanded message is *N* x 1024 bits.

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

These registers are initialized to the following 64-bit integers (hexadecimal values):

a = 6A09E667F3BCC908
b = BB67AE8584CAA73B
c = 3C6EF372FE94F82B
c = A54FF53A5F1D36F1
e = 510E527FADE682D1
f = 9B05688C2B3E6C1F
g = 1F83D9ABFB41BD6B

h = 5BE0CDI9137E2179

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

Step 4: Process message in 1024-bit (128-word) blocks. The heart of the algorithm is a module that consists of 80 rounds; this module is labeled F in Figure 1. The logic is illustrated in Figure 2.

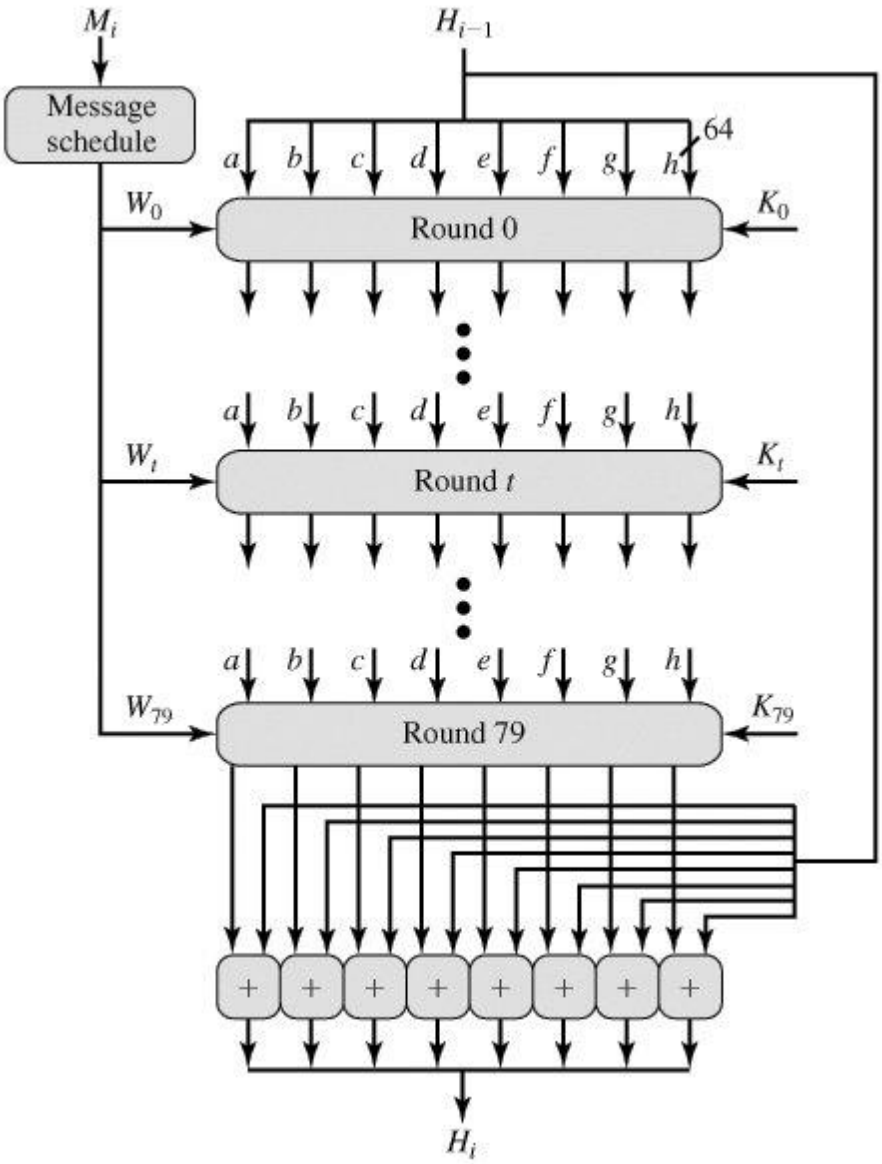


Figure 2: SHA-512 Processing of a Single 1024-Bit Block

Each round takes as input the 512-bit buffer value abcdefgh, and updates the contents of the buffer. At input to the first round, the buffer has the value of the intermediate hash value, H_{i-1}.

Each round t makes use of a 64-bit value Wt derived from the current 1024-bit block being processed (Mi) these values are derived using a message schedule described subsequently.

Each round also makes use of an additive constant Kt where 0≤ t ≤79 indicates one of the 80 rounds. These words represent the first sixty-four bits of the fractional parts of the cube roots of the first eighty prime numbers. The constants provide a "randomized" set of 64-bit patterns, which should eliminate any regularities in the input data.

The output of the eightieth round is added to the input to the first round (H_{i-1}) to produce H_i. The addition is done independently for each of the eight words in the buffer with each of the corresponding words in H_{i-1} using addition modulo 2^{64}.

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

We can summarize the behavior of SHA-512 as follows:

*H*0 = IV

Hi = SUM64(*Hi*-1, abcdefgh$_i$)

MD = *HN*

Where

IV = initial value of the abcdefgh buffer, defined in step 3

abcdefgh$_i$ = the output of the last round of processing of the *i*th message block

N = the number of blocks in the message (including padding and length fields)

SUM_{64} = Addition modulo 2^{64} performed separately on each word of the pair of inputs

MD = final message digest value

SHA-512 Round Function

The logic in each of the 80 steps of the processing of one 512-bit block. Each round is defined by the following set of equations:

$$T_1 = h + \text{Ch}(e, f, g) + \left(\sum_1^{512} e\right) + W_t + K_t$$

$$T_2 = \left(\sum_0^{512} a\right) + \text{Maj}(a, b, c)$$

$$a = T_1 + T_2$$

$$b = a$$

$$c = b$$

$$d = c$$

$$e = d + T_1$$

$$f = e$$

$$g = f$$

$$h = g$$

Where

t =step number; 0 ≤*t*≤ 79

Ch(*e*, *f*, *g*)= (*e* AND *f*) XOR (NOT *e* AND *g*) *the conditional function*: *If e* then *f* else *g*

Maj(*a*, *b*, *c*) = (*a* AND *b*) XOR (*a* AND *c*) XOR (*b* AND *c*) *the function is true only of the majority (two or three) of the arguments are true.*

$$\left(\sum_0^{512} a\right) = \text{ROTR}^{28}(a) \oplus \text{ROTR}^{34}(a) \oplus \text{ROTR}^{39}(a)$$

$$\left(\sum_1^{512} e\right) = \text{ROTR}^{14}(e) \oplus \text{ROTR}^{18}(e) \oplus \text{ROTR}^{41}(e)$$

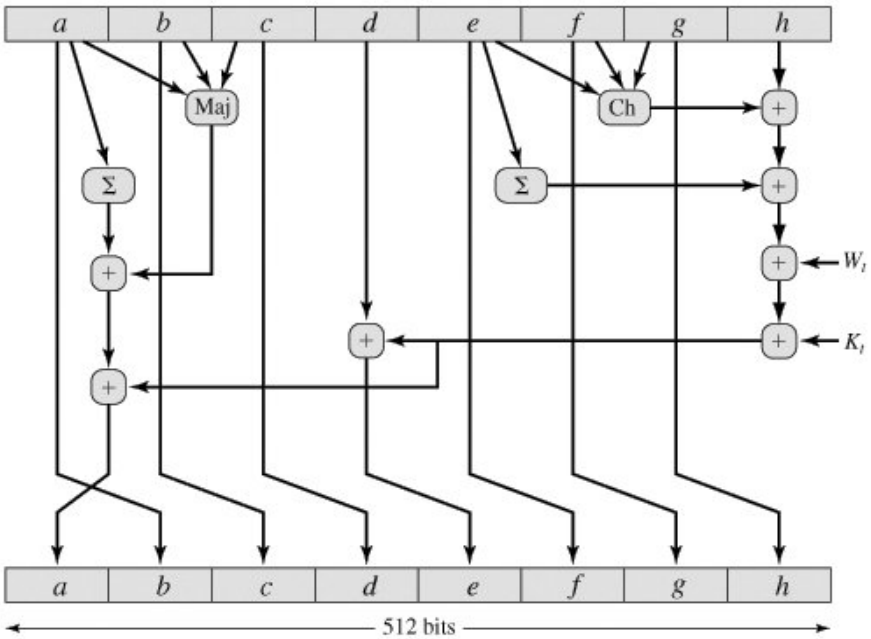
$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument *x* by *n* bits

W_t = a 64-bit word derived from the current 512-bit input block

K_t = a 64-bit additive constant

+ = addition modulo 2^{64}

Figure 3. Elementary SHA-512 Operation (single round)



It remains to indicate how the 64-bit word values *Wt* are derived from the 1024-bit message. Figure 4 illustrates the mapping. The first 16 values of *Wt* are taken directly from the 16 words of the current block. The remaining values are defined as follows:

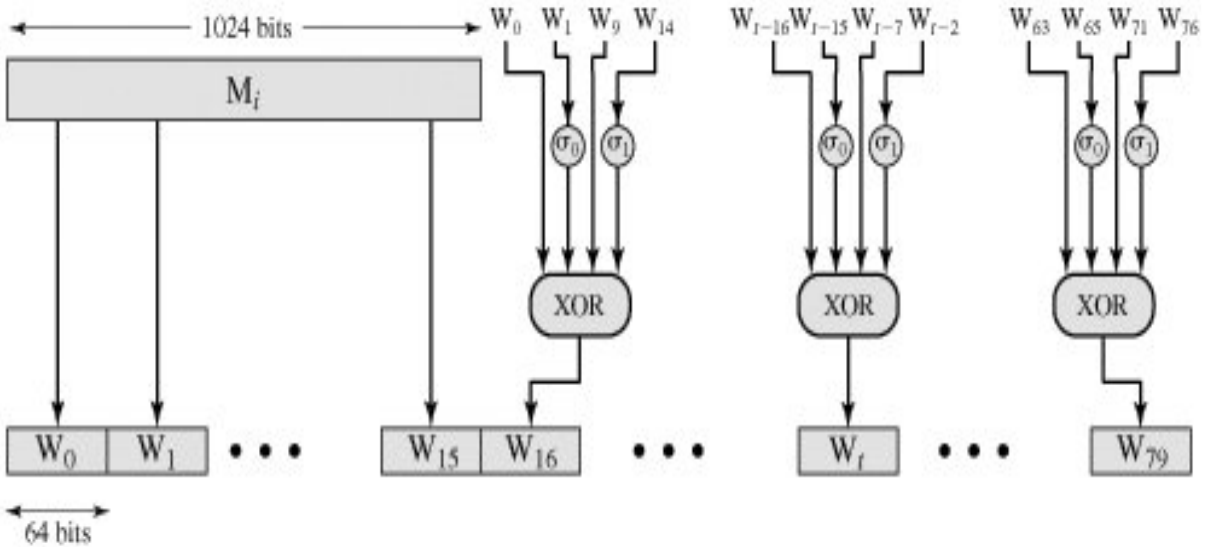
$$W_t = \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16}$$

Where,

$$\sigma_0^{512}(x) = \text{ROTR}^1(x) \oplus \text{ROTR}^8(x) \oplus \text{SHR}^7(x)$$
$$\sigma_1^{512}(x) = \text{ROTR}^{19}(x) \oplus \text{ROTR}^{61}(x) \oplus \text{SHR}^6(x)$$

$\text{ROTR}^n(x)$ = circular right shift (rotation) of the 64-bit argument *x* by *n* bits
$\text{SHR}^n(x)$ = left shift of the 64-bit argument *x* by *n* bits with padding *by zeros on the right*

Figure 4. Creation of 80-word Input Sequence for SHA-512 processing of Single Block



Thus, in the first 16 steps of processing, the value of *Wt* is equal to the corresponding word in the message block. For the remaining 64 steps, the value of *Wt* consists of the circular left shift by one bit of the XOR of four of the preceding values of *Wt*, with two of those values subjected to shift and rotate operations. This introduces a great deal of redundancy and interdependence into the message blocks that are compressed, which complicates the task of finding a different message block that maps to the same compression function output.

Whirlpool Secure Hash Function

- We examine the hash function **Whirlpool**, one of whose designers is also co inventor of **Rijndael**, adopted as the Advanced Encryption Standard (AES).
- Whirlpool is one of only two hash functions endorsed by NESSIE (New European Schemes for Signatures, Integrity, and Encryption).
- The NESSIE project is a European Union sponsored effort to put forward a portfolio of strong cryptographic primitives of various types.
- Whirlpool is based on the use of a block cipher for the compression function.

Whirlpool is a block-cipher based hash function intended to provide security and performance that is comparable, if not better, than that found in non-block-cipher based hash functions, such as SHA.

Whirlpool has the following features:

1. The hash code length is 512 bits, equaling the longest hash code available with SHA.
2. The overall structure of the hash function is one that has been shown to be resistant to the usual attacks on block-cipher-based hash codes.
3. The underlying block cipher is based on AES and is designed to provide for implementation in both software and hardware that is both compact and exhibits good performance.

Whirlpool Hash Structure:

Whirlpool Logic

Given a message consisting of a sequence of blocks m1;m2; . . .;mt, the Whirlpool hash function is expressed as follows:

$$H_0 = \text{initial value.}$$

$$H_i = E(H_{i-1}, m_i) \oplus H_{i-1} \oplus m_i = \text{intermediate value.}$$

$$H_t = \text{hash code value.}$$

The encryption key input for each iteration i is the intermediate hash H_{i-1} value from the previous iteration, and the plaintext is the current message block m_i.

The output for this iteration (H_i) consists of the bitwise XOR of the current message block, the intermediate hash value from the previous iteration, and the output from W.

Whirlpool secure hash function:-

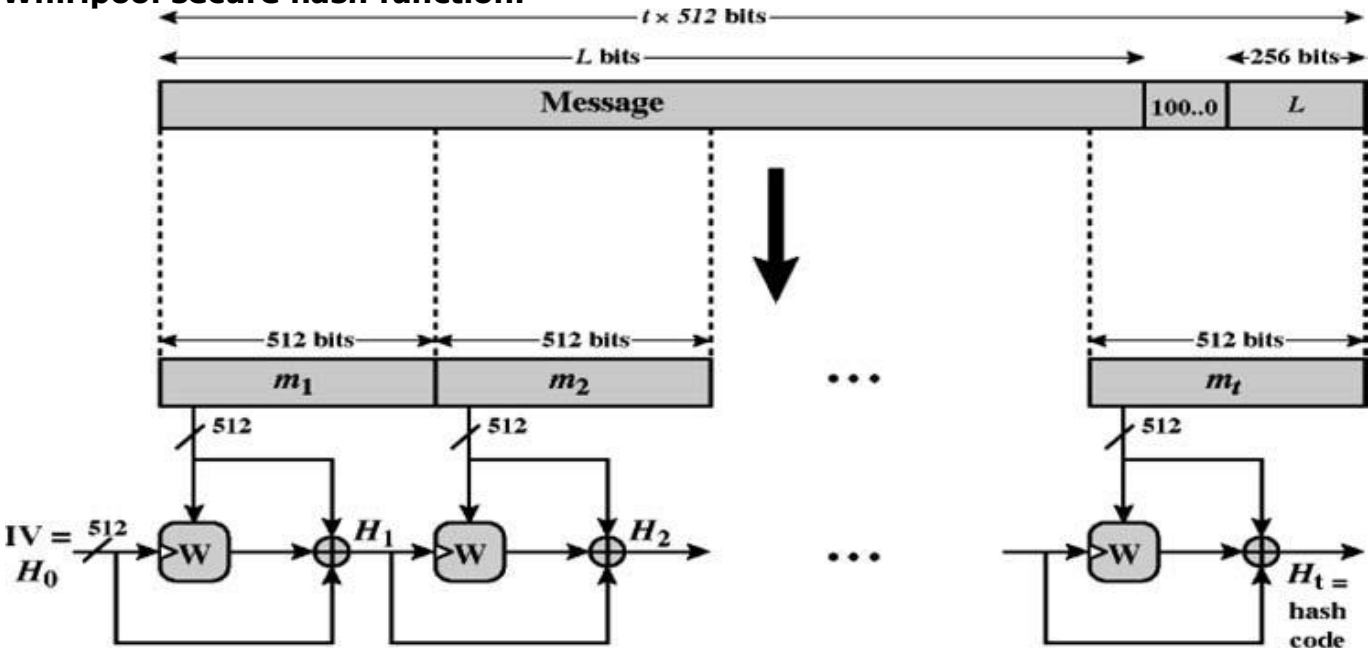


Figure 1. Message digest generation using Whirlpool. Note: triangular hatch marks key input.

The algorithm takes as input a message with a maximum length of less than 2^{256} bits and produces as output a 512-bit message digest. The input is processed in 512-bit blocks. Figure 1 depicts the overall processing of a message to produce a digest. The processing consists of the following steps:

Step 1: Append padding bits. The message is padded so that its length in bits is an odd multiple of 256. Padding is always added, even if the message is already of the desired length.

For example, if the message is 256 X 3=768 bits long, it is padded by 512 bits to a length of 256X 5 = 1280 bits. Thus, the number of padding bits is in the range of 1 to 512.

The padding consists of a single 1-bit followed by the necessary number of 0-bits.

Step 2: Append length. A block of 256 bits is appended to the message. This block is treated as an unsigned 256-bit integer (most significant byte first) and contains the length in bits of the original message (before the padding).

The outcome of the first two steps yields a message that is an integer multiple of 512 bits in length. In Figure 1, the expanded message is represented as the sequence of 512-bit blocks m1,m2, . . . mt, so that the total length of the expanded message is tX512 bits. These blocks are viewed externally as arrays of bytes by sequentially grouping the bits in 8-bit chunks. However, internally, the hash state Hi is viewed as an 8 X 8 matrix of bytes. The transformation between the two is explained subsequently.

Step 3: Initialize hash matrix. An 8 X 8 matrix of bytes is used to hold intermediate and final results of the hash function. The matrix is initialized as consisting of all 0-bits.

Step 4: Process message in 512-bit (64-byte) blocks. The heart of the algorithm is the block cipher W.

Block Cipher W

Unlike virtually all other proposals for a block-cipher-based hash function, Whirlpool uses a block cipher that is specifically designed for use in the hash function and that is unlikely ever to be used as a standalone encryption function. The reason for this is that the designers wanted to make use of a block cipher with the security and efficiency of AES but with a hash length that provided a potential security equal to SHA-512. The result is the block cipher W, which has a similar structure and uses the same elementary functions as AES, but which uses a block size and a key size of 512 bits. Table 1 compares AES and W.

Table 1. Comparison of Whirlpool block cipher W and AES

	W	AES
Block size (bits)	512	128
Key size (bits)	512	128, 192, or 256
Matrix orientation	input is mapped row-wise	Input is mapped column-wise
Number of rounds	10	10, 12, or 14
Key expansion	W round function	dedicated expansion algorithm
$GF(2^8)$ polynomial	$x^8 + x^4 + x^3 + x^2 + 1$ (011D)	$x^8 + x^4 + x^3 + x + 1$ (011B)
Origin of S-box	recursive structure	multiplicative inverse in $GF(2^8)$ plus affine transformation
Origin of round constants	successive entries of the S-box	elements 2^i of $GF(2^8)$
Diffusion layer	right multiplication by 8×8 circulant MDS matrix (1, 1, 4, 1, 8, 5, 2, 9) - mix rows	left multiplication by 4×4 circulant MDS matrix (2, 3, 1, 1) - mix columns
Permutation	shift columns	shift rows

Although W is similar to AES, it is not simply an extension. In fact, AES is one version of the cipher Rijndael, which was submitted as a candidate for the AES. The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192, or 256 bits.

The AES specification uses the same three key size alternatives but limits the block length to 128 bits. AES operates on a state of 4 X 4 bytes. Rijndael with block length 192 bits operates on a state of 4 X 6 bytes. Rijndael with block length 256 bits operates on a state of 4 X 8 bytes. W operates on a state of 8 X8 bytes.

The more the state representation differs from a square, the slower the diffusion goes and the more rounds the cipher needs. For a block length of 512 bits, the Whirlpool developers could have defined a Rijndael operating on a state of 4 X16 bytes, but that cipher would have needed many rounds and it would have been very slow.

As Table 1 indicates, W uses a row-oriented matrix whereas AES uses a column-oriented matrix. There is no technical reason to prefer one orientation to another, because one can easily construct an equivalent description of the same cipher, exchanging rows with columns.

Figure 2 shows the overall structure of W. The encryption algorithm takes a 512-bit block of plaintext and a 512-bit key as input and produces a 512-bit block of cipher text as output. The encryption algorithm involves the use of four different functions or transformations: add key (AK), substitute bytes (SB), shift columns (SC), and mix rows (MR), whose operations are explained subsequently. W consists of a single application of AK followed by 10 rounds that involve all four functions. We can concisely express the operation of a round r as a round function RF that is a composition of functions:

$$RF(\mathbf{K_r}) = \mathbf{AK}[\mathbf{K_r}] \circ \mathbf{MR} \circ \mathbf{SC} \circ \mathbf{SB}$$

Where K_r is the round key matrix for round r.
The overall algorithm, with key input K, can be defined as follows

$$|W(\mathbf{K}) = \left(\bigcirc_{r=1}^{10} RF(\mathbf{K}_r) \right) \circ AK(\mathbf{K}_0)$$

Where the large circle indicates iteration of the composition function with index r running from 1 through 10.

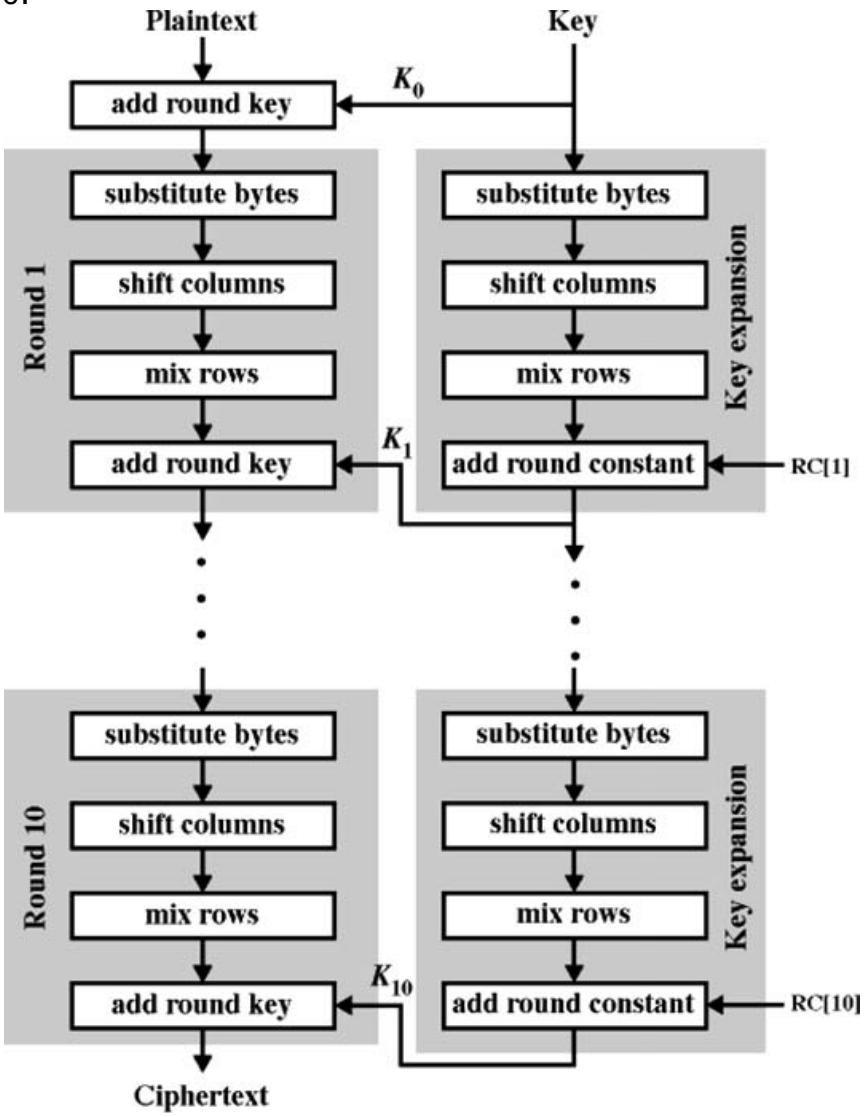


Figure 2. Whirlpool cipher W.

The plaintext input to W is a single 512-bit block. This block is treated as an 8 X 8 square matrix of bytes, labeled CState. Figure 3 illustrates that the ordering of bytes within a matrix is by row. So, for example, the first eight bytes of a 512-bit plaintext input to the encryption cipher occupy the first row of the internal matrix CState, the second eight bytes occupy the second row, and so on.

The representation of the linear byte stream as a square matrix can be concisely expressed as a mapping function μ. For a linear byte array X with elements X_k ($0 \le k \le 63$), the corresponding matrix A with elements $a_{i,j}$ ($0 \le i; j \le 7$), we have the following correspondence:

$$\mathbf{A} = \mu(X) \Leftrightarrow a_{i,j} = x_{8i+j}.$$

Similarly, the 512-bit key is depicted as a square matrix KState of bytes. This key is used as input to the initial AK function. The key is also expanded into a set of 11 round keys, as explained subsequently.

We now look at the individual functions that are part of W.

The Nonlinear Layer SB

The SB function can be expressed by the following correspondence, for an input matrix A and an output matrix B:

$$\mathbf{B} = SB(\mathbf{A}) \Leftrightarrow b_{i,j} = S[a_{i,j}], \quad 0 \le i, j \le 7.$$

Where S[x]refers to the mapping of input byte x into output byte S[x] by the S-box.

The Permutation Layer SC

The permutation layer (shift columns) causes a circular downward shift of each column of CState except the first column. The SC function can be expressed by the following correspondence, for an input matrix A and an output matrix B:

$$\mathbf{B} = SC(\mathbf{A}) \Leftrightarrow b_{i,j} = a_{(i-j) \bmod 8, j} \quad 0 \le i, j \le 7.$$

The Diffusion Layer MR

Diffusion is a cryptographic property introduced by Claude Shannon. By diffusion, Shannon meant spreading out the influence of a single plaintext digit over many cipher text digits so as to hide the statistical structure of the plaintext.

The transformation can be defined by the matrix multiplication: B = AC, where A is the input matrix, B is the output matrix, and C is the transformation matrix:

$$C = \begin{bmatrix} 01 & 01 & 04 & 01 & 08 & 05 & 02 & 09 \\ 09 & 01 & 01 & 04 & 01 & 08 & 05 & 02 \\ 02 & 09 & 01 & 01 & 04 & 01 & 08 & 05 \\ 05 & 02 & 09 & 01 & 01 & 04 & 01 & 08 \\ 08 & 05 & 02 & 09 & 01 & 01 & 04 & 01 \\ 01 & 08 & 05 & 02 & 09 & 01 & 01 & 04 \\ 04 & 01 & 08 & 05 & 02 & 09 & 01 & 01 \\ 01 & 04 & 01 & 08 & 05 & 02 & 09 & 01 \end{bmatrix}$$

Each element in the product matrix is the sum of products of elements of one row and one column. In this case, the individual additions and multiplications are performed in GF (2^8)

with the irreducible polynomial $f(x) = x^8 + x^4 + x^3 + x^2 + 1$. As an example of the matrix multiplication involved, the first element of the output matrix is:

$$b_{0,0} = a_{0,0} \oplus (9 \bullet a_{0,1}) \oplus (2 \bullet a_{0,2}) \oplus (5 \bullet a_{0,3}) \oplus (8 \bullet a_{0,4}) \oplus a_{0,5} \oplus (4 \bullet a_{0,6}) \oplus a_{0,7}.$$

The Add Key Layer AK

In the add key layer, the 512 bits of CState are bitwise XORed with the 512 bits of the round key. The AK function can be expressed by the following correspondence, for an input matrix A, an output matrix B, and a round key Ki:

$$\mathbf{B} = AK[K_i](\mathbf{A}) \Leftrightarrow b_{i,j} = a_{i,j} \oplus k_{i,j}, \quad 0 \leq i, j \leq 7.$$

HMAC Algorithm

- **HMAC algorithm** stands for Hashed or Hash-based Message Authentication Code. It is a result of work done on developing a MAC derived from cryptographic hash functions.
- HMAC is a great resistance towards cryptanalysis attacks as it uses the Hashing concept twice. HMAC consists of twin benefits of Hashing and MAC and thus is more secure than any other authentication code.
- RFC 2104 has issued HMAC, and HMAC has been made compulsory to implement in IP security.
- The FIPS 198 NIST standard has also issued HMAC.

Objectives –

- As the Hash Function, HMAC is also aimed to be one way, i.e, easy to generate output from input but complex the other way round.
- It aims at being less affected by collisions than the hash functions.
- HMAC reuses the algorithms like MD5 and SHA-1 and checks to replace the embedded hash functions with more secure hash functions, in case found.
- HMAC tries to handle the Keys in a more simple manner.

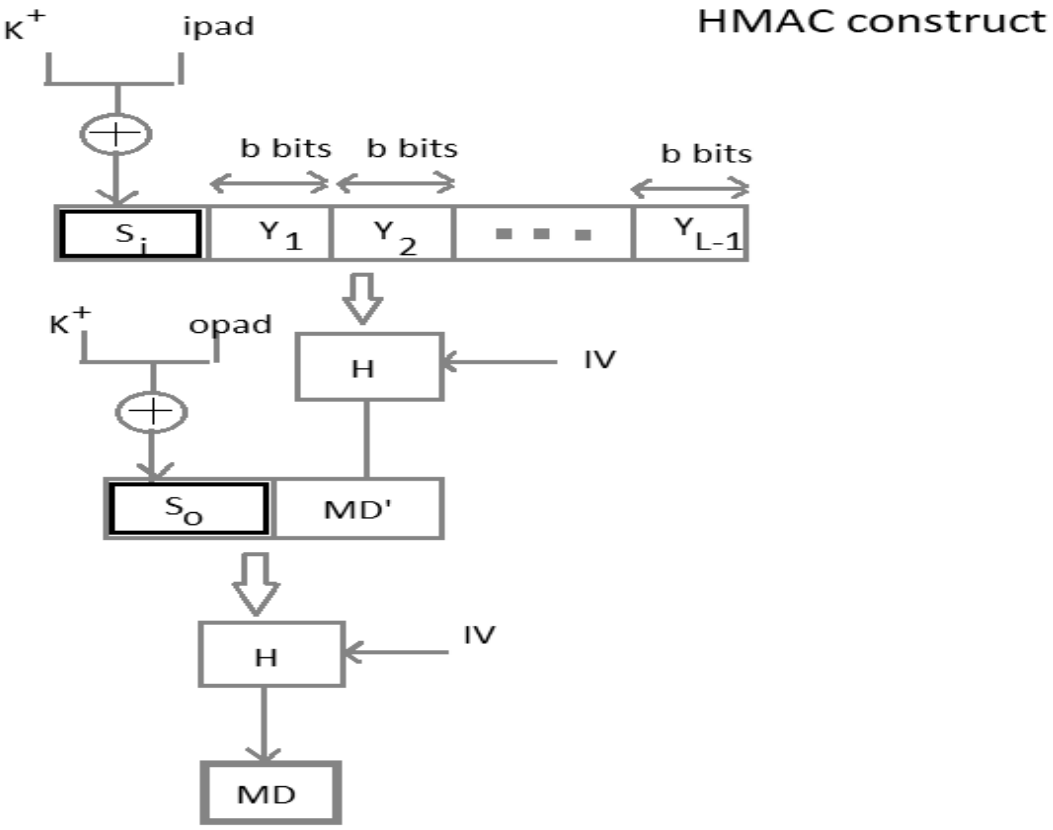
HMAC algorithm –

The working of HMAC starts with taking a message M containing blocks of length *b* bits. An input signature is padded to the left of the message and the whole is given as input to a hash function which gives us a temporary message-digest MD'. MD' again is appended to an output signature and the whole is applied a hash function again, the result is our final message digest MD.

Here is a simple structure of HMAC:

Here, H stands for Hashing function,
M is the original message
Si and So are input and output signatures respectively,
Yi is the ith block in original message M, where I ranges from [1, L]
L = the count of blocks in M

K is the secret key used for hashing
IV is an initial vector (some constant)
The generation of input signature and output signature *Si* and *So* respectively.



$S_i = K^+ \oplus ipad$ where K^+ is nothing but K padded with zeros on the left so that the result is b bits in length

$S_o = K^+ \oplus opad$ where ipad and opad are 00110110 and 01011100 respectively taken b/8 times repeatedly.

$MD' = H(S_i || M)$

$MD = H(S_o || MD')$ or $MD = H(S_o || H(S_i || M))$

To a normal hash function, HMAC adds a compression instance to the processing. This structural implementation holds efficiency for shorter MAC values.

Working of HMAC

HMAC provides client and server with a shared private key that is known only to them. The client makes a unique hash (HMAC) for every request. When the client requests the server, it hashes the requested data with a private key and sends it as a part of the request. Both the message and key are hashed in separate steps making it secure. When the server receives the request, it makes its own HMAC. Both the HMACS are compared and if both are equal, the client is considered legitimate.

The formula **for HMAC:**
HMAC = hashFunc(secret key + message)

There are three types of authentication functions. They are message encryption, message authentication code, and hash functions. The major difference between MAC and hash (HMAC here) is the dependence of a key. In HMAC we have to apply the hash function along with a key on the plain text. The hash function will be applied to the plain text message. But before applying, we have to compute S bits and then append it to plain text and after that apply the hash function. For generating those S bits we make use of a key that is shared between the sender and receiver.

Using key K ($0 < K < b$), K+ is generated by padding O's on left side of key K until length becomes b bits. The reason why it's not padded on right is change(increase) in the length of key. b bits because it is the block size of plain text. There are two predefined padding bits called ipad and opad. All this is done before applying hash function to the plain text message.

ipad - 00110110

opad – 01011100

Now we have to calculate S bits

K+ is EXORed with ipad and the result is S1 bits which is equivalent to b bits since both K+ and ipad are b bits. We have to append S1 with plain text messages. Let P be the plain text message.
S1, p0, p1 upto Pm each is b bits. m is the number of plain text blocks. P0 is plain text block and b is plain text block size. After appending S1 to Plain text we have to apply HASH algorithm (any variant). Simultaneously we have to apply initialization vector (IV) which is a buffer of size n-bits. The result produced is therefore n-bit hash code i.e H(S1 || M).
Similarly, n-bits are padded to b-bits And K+ is EXORed with opad producing output S2 bits. S2 is appended to the b-bits and once again hash function is applied with IV to the block. This further results into n-bit hash code which is H(S2 || H(S1 || M)).

Advantages

- HMACs are ideal for high-performance systems like routers due to the use of hash functions which are calculated and verified quickly unlike the public key systems.
- Digital signatures are larger than HMACs, yet the HMACs provide comparably higher security.
- HMACs are used in administrations where public key systems are prohibited.

Disadvantages

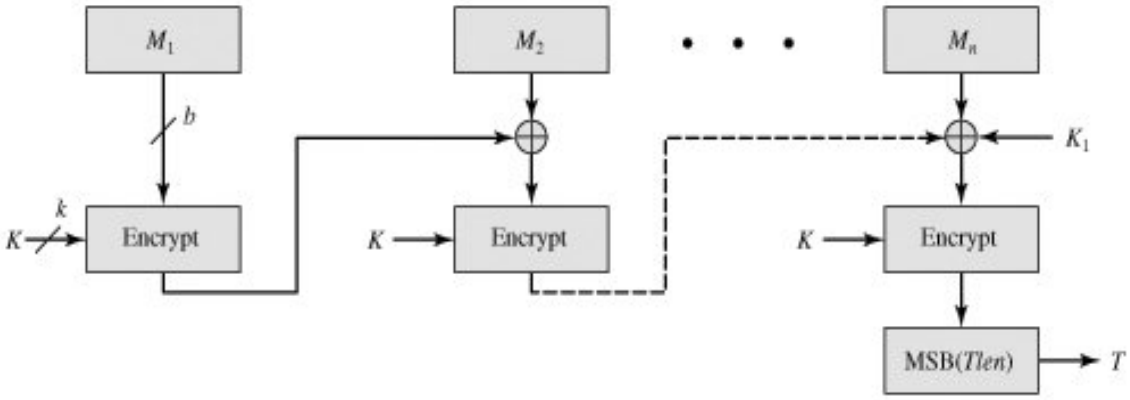
- HMACs uses shared key which may lead to non-repudiation. If either sender or receiver's key is compromised then it will be easy for attackers to create unauthorized messages.

CMAC Algorithm

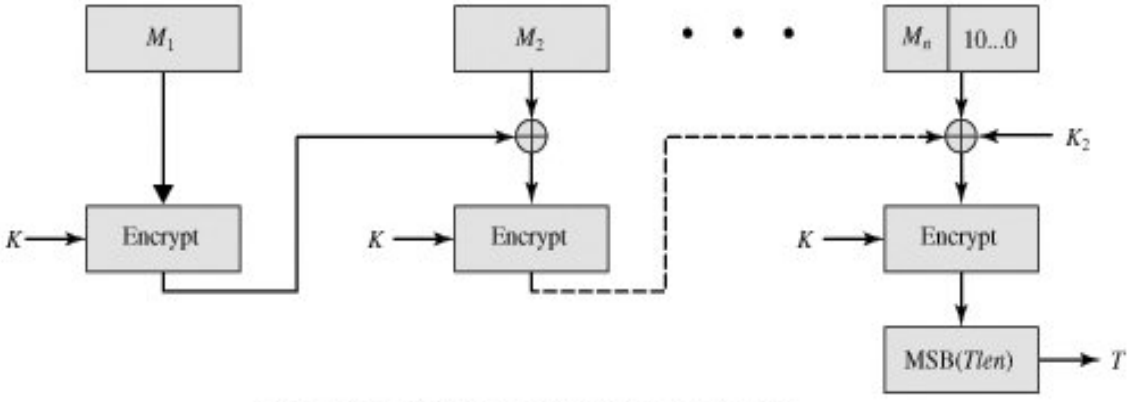
The Data Authentication Algorithm defined in FIPS PUB 113, also known as the CBC-MAC (cipher block chaining message authentication code). This cipher-based MAC has been widely adopted in government and industry.MAC is secure under a reasonable set of security criteria, with the following restriction.

Only messages of one fixed length of mn bits are processed, where n is the cipher block size and m is a fixed positive integer. Black and Rogaway demonstrated that this limitation could be overcome using three keys: one key of length k to be used at each step of the cipher block chaining and two keys of length n, where k is the key length and n is the cipher block length. This proposed construction was refined by Iwata and Kurosawa so that the two n-bit keys could be derived from the encryption key, rather than being provided separately.

First, let us consider the operation of CMAC when the message is an integer multiple n of the cipher block length b. For AES, b = 128 and for triple DES, b = 64. The message is divided into n blocks, M_1, M_2... Mn. The algorithm makes use of a k-bit encryption key K and an n-bit constant K_1. For AES, the key size k is 128, 192, or 256 bits; for triple DES, the key size is 112 or 168 bits. CMAC is calculated as follows:



(a) Message length is integer multiple of block size



(b) Message length is not integer multiple of block size

$$C_1 = E(K, M_1)$$

$$C_2 = E(K, [M_2 \oplus C_1])$$

$$C_3 = E(K, [M_3 \oplus C_2])$$

.

.

.

$$C_n = E(K, [M_n \oplus C_{n1} \oplus K_1])$$

$$T = MSB_{Tlen}(C_n)$$

Where

T = message authentication code, also referred to as the tag
T_{len} = bit length of T
MSBs(X) = the s leftmost bits of the bit string X

If the message is not an integer multiple of the cipher block length, then the final block is padded to the right (least significant bits) with a 1 and as many 0s as necessary so that the final block is also of length b. The CMAC operation then proceeds as before, except that a different n-bit key $K2$ is used instead of $K1$.
The two n-bit keys are derived from the k-bit encryption key as follows:
$L = E(K, 0n)$
$K1 = L \cdot x$
$K2 = L \cdot x2 = (L \cdot x) \cdot x$

Where multiplication ($\cdot$) is done in the finite field (2^n) and x and x^2 are first and second order polynomials that are elements of GF($2n$) Thus the binary representation of x consists of n - 2 zeros followed by 10; the binary representation of $x2$ consists of n - 3 zeros followed by 100.

The finite field is defined with respect to an irreducible polynomial that is lexicographically first among all such polynomials with the minimum possible number of nonzero terms. For the two approved block sizes, the polynomials are and $x^{64} + x^4 + x^3 + x + 1$ and $x^{128} + x^7 + x^2 + x + 1$.

To generate $K1$ and $K2$ the block cipher is applied to the block that consists entirely of 0 bits. The first sub key is derived from the resulting cipher text by a left shift of one bit, and, conditionally, by XORing a constant that depends on the block size. The second sub key is derived in the same manner from the first sub key.

Digital Signature

A digital signature is a mathematical technique used to validate the authenticity and integrity of a message, software, or digital document.

1. **Key Generation Algorithms**: Digital signature is electronic signatures, which assure that the message was sent by a particular sender. While performing digital transactions authenticity and integrity should be assured, otherwise, the data can be altered or someone can also act as if he was the sender and expect a reply.

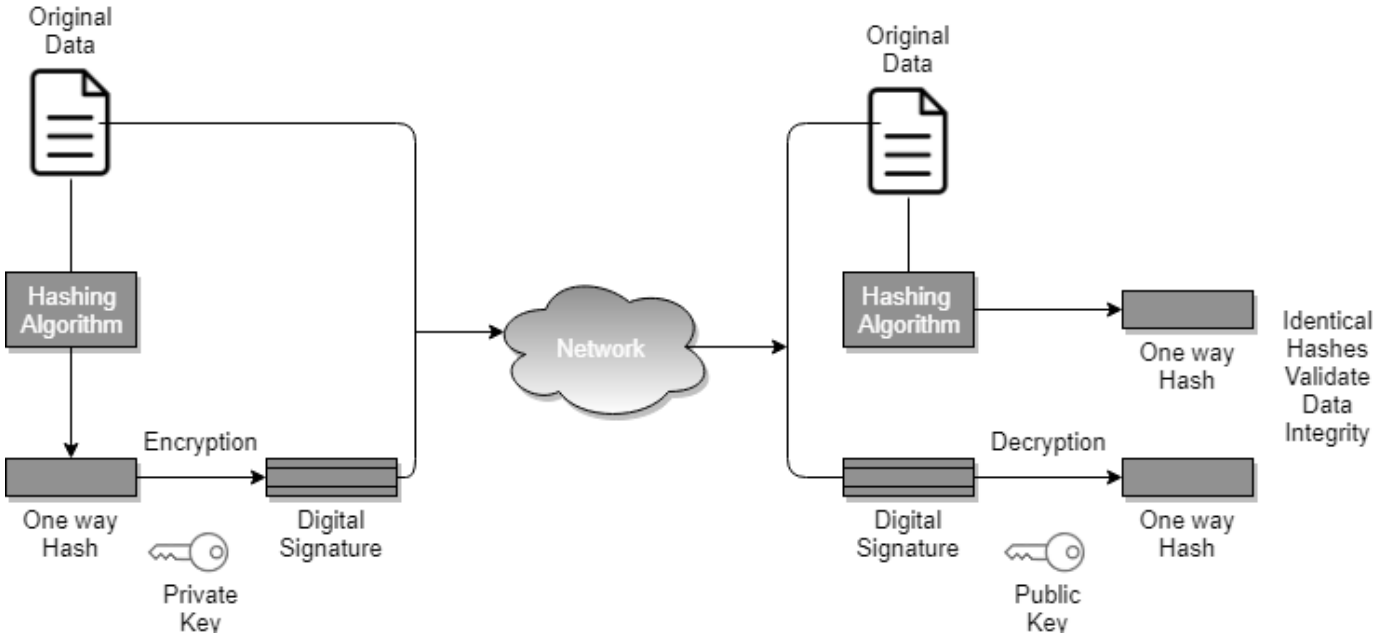
2. **Signing Algorithms**: To create a digital signature, signing algorithms like email programs create a one-way hash of the electronic data which is to be signed. The signing algorithm then encrypts the hash value using the private key (signature key). This encrypted hash along with other information like the hashing algorithm is the digital signature. This digital signature is appended with the data and sent to the verifier. The reason for encrypting the hash instead of the entire message or document is that a hash function converts any arbitrary input into a much shorter fixed-length value. This saves time as now instead of signing a long message a shorter hash value has to be signed and moreover hashing is much faster than signing.

3. **Signature Verification Algorithms**: Verifier receives Digital Signature along with the data. It then uses Verification algorithm to process on the digital signature and the public key (verification key) and generates some value. It also applies the same hash function on the received data and generates a hash value. Then the hash value and the output of the verification algorithm are compared. If they both are equal, then the digital signature is valid else it is invalid.

The steps followed in creating digital signature are:

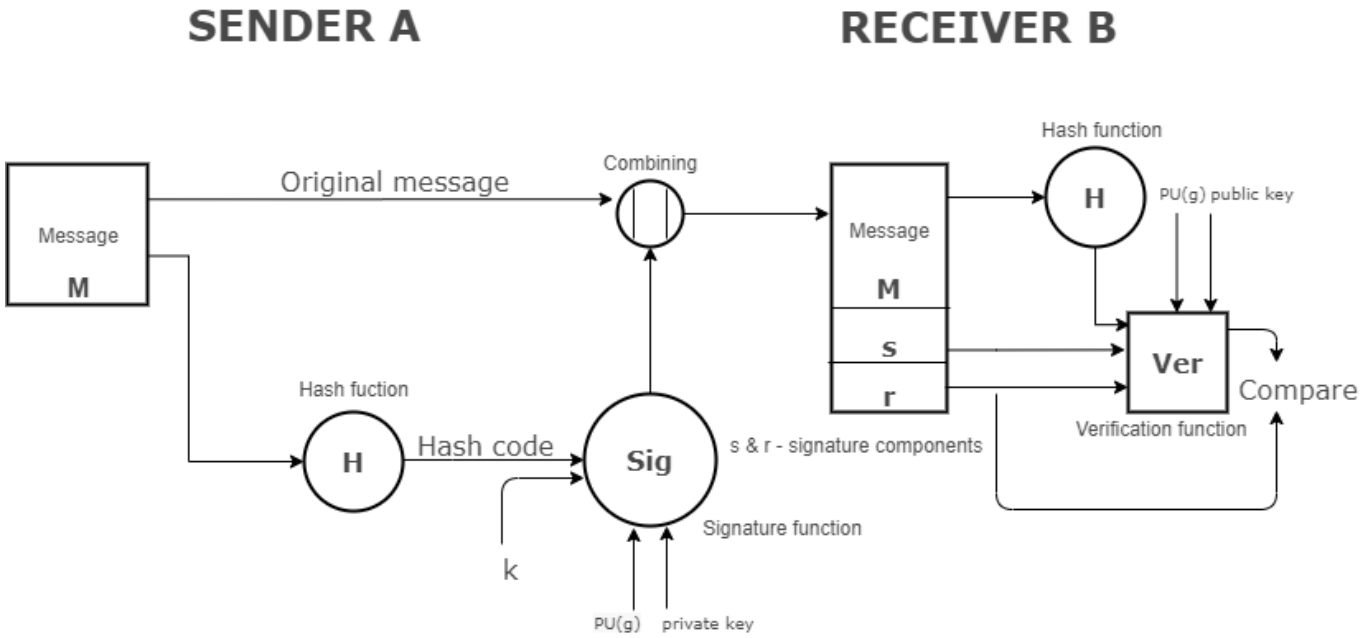
1. Message digest is computed by applying hash function on the message and then message digest is encrypted using private key of sender to form the digital signature. (digital signature = encryption (private key of sender, message digest) and message digest = message digest algorithm(message)).
2. Digital signature is then transmitted with the message.(message + digital signature is transmitted)
3. Receiver decrypts the digital signature using the public key of sender.(This assures authenticity, as only sender has his private key so only sender can encrypt using his private key which can thus be decrypted by sender's public key).
4. The receiver now has the message digest.
5. The receiver can compute the message digest from the message (actual message is sent with the digital signature).
6. The message digest computed by receiver and the message digest (got by decryption on digital signature) need to be same for ensuring integrity.

Message digest is computed using one-way hash function, i.e. a hash function in which computation of hash value of a message is easy but computation of the message from hash value of the message is very difficult.



Digital Signature Standard (DSS)

Digital Signature Standard (DSS) is a Federal Information Processing Standard (FIPS) which defines algorithms that are used to generate digital signatures with the help of Secure Hash Algorithm (SHA) for the authentication of electronic documents. DSS only provides us with the digital signature function and not with any encryption or key exchanging strategies.



Sender Side:

In DSS Approach, a hash code is generated out of the message and following inputs are given to the signature function –

1. The hash code.
2. The random number 'k' generated for that particular signature.

3. The private key of the sender i.e., PR(a).
4. A global public key(which is a set of parameters for the communicating principles) i.e., PU(g).

These input to the function will provide us with the output signature containing two components – 's' and 'r'. Therefore, the original message concatenated with the signature is sent to the receiver.

Receiver Side:

At the receiver end, verification of the sender is done. The hash code of the sent message is generated. There is a verification function which takes the following inputs –

1. The hash code generated by the receiver.
2. Signature components 's' and 'r'.
3. Public key of the sender.
4. Global public key.

The output of the verification function is compared with the signature component 'r'. Both the values will match if the sent signature is valid because only the sender with the help of it private key can generate a valid signature.

DSA Algorithm (Digital Signature Algorithm)

The first part of the DSA algorithm is the public key and private key generation through some steps, which can be told as:

- Firstly, choose a prime number q, which is called the prime divisor in this.
- Then, choose another primer number p, such that p-1 mod q = 0. p is called the prime modulus in this.
- Then, choose an integer g, such that 1 < g < p, g**q mod p = 1 and g = h**((p–1)/q) mod p. q is also called g's multiplicative order modulo p in this algorithm.
- Then, choose an integer, such that 0 < x < q for this.
- Now, compute y as g**x mod p.
- Thus, Package the public key as {p,q,g,y} is this.
- And, Package the private key as {p,q,g,x} is this.

Then, the second part of the DSA algorithm is the signature generation and signature verification in this algorithm, which can be told as:

Firstly, to generate a message signature, the sender can follow these further steps:

- Firstly, generate the message digest h, using a hash algorithm like SHA1.
- Then, generate a random number k, such that 0 < k < q.
- Then, Computer as (g**k mod p) mod q. If r = 0, select a different k.
- And, Compute i, such that k*i mod q = 1. i is called the modular multiplicative inverse of k modulo q in this.
- Then, Compute s = i*(h+r*x) mod q. If s = 0, select a different k.
- Thus, Package the digital signature as {r,s}.

Then, to verify a message signature, the receiver of the message and the digital signature can follow these further steps as:

- Firstly, Generate the message digest h, using the same hash algorithm.
- Then, Compute w, such that s*w mod q = 1. w is called the modular multiplicative inverse of s modulo q in this.
- Then, Compute u1 = h*w mod q.
- And, Compute u2 = r*w mod q.
- Then, Compute v = (((g**u1)*(y**u2)) mod p) mod q.
- Wherever, if v == r, the digital signature is valid.

Knapsack Encryption Algorithm

Knapsack Encryption Algorithm is the first general public key cryptography algorithm. It is developed by **Ralph Merkle** and **Mertin Hellman** in 1978. As it is a Public key cryptography, it needs two different keys. One is Public key which is used for Encryption process and the other one is Private key which is used for Decryption process. In this algorithm we will two different knapsack problems in which one is easy and other one is hard. The easy knapsack is used as the private key and the hard knapsack is used as the public key. The easy knapsack is used to derived the hard knapsack.
For the easy knapsack, we will choose a **Super Increasing knapsack problem**. Super increasing knapsack is a sequence in which every next term is greater than the sum of all preceding terms.

Example –

{1, 2, 4, 10, 20, 40} is a super increasing as

1<2, 1+2<4, 1+2+4<10, 1+2+4+10<20 and 1+2+4+10+20<40.

Derive the Public key

Step-1:

Choose a super increasing knapsack {1, 2, 4, 10, 20, 40} as the private key.

Step-2:

Choose two numbers n and m. Multiply all the values of private key by the number n and then find modulo m. The value of m must be greater than the sum of all values in private key, for example 110. And the number n should have no common factor with m, for example 31.

Step-3:

Calculate the values of Public key using m and n.

1x31 mod(110) = 31

2x31 mod(110) = 62

4x31 mod(110) = 14

10x31 mod(110) = 90

20x31 mod(110) = 70

40x31 mod(110) = 30

Thus, our public key is {31, 62, 14, 90, 70, 30}
And Private key is {1, 2, 4, 10, 20, 40}.

Now take an example for understanding the process of encryption and decryption.

Example –
Lets our plain text is 100100111100101110.

1. Encryption:
As our knapsacks contain six values, so we will split our plain text in a groups of six:
100100 111100 101110

Multiply each values of public key with the corresponding values of each group and take their sum.

100100 {31, 62, 14, 90, 70, 30}

1x31+0x62+0x14+1x90+0x70+0x30 = 121

111100 {31, 62, 14, 90, 70, 30}

1x31+1x62+1x14+1x90+0x70+0x30 = 197

101110 {31, 62, 14, 90, 70, 30}

1x31+0x62+1x14+1x90+1x70+0x30 = 205

So, our cipher text is 121 197 205.

2. Decryption :
The receiver receive the cipher text which has to be decrypt. The receiver also knows the values of m and n.
So, first we need to find the n^{-1}, which is multiplicative inverse of n mod m i.e.,

$n \times n^{-1} \bmod m=1$

$31 \times n^{-1} \bmod (110)=1$

$n^{-1} =71$

Now, we have to multiply 71 with each block of cipher text take modulo m.

121 x 71 mod(110) = 11

Then, we will have to make the sum of 11 from the values of private key {1, 2, 4, 10, 20, 40} i.e.,
1+10=11 so make that corresponding bits 1 and others 0 which is 100100.
Similarly,

197 x 71 mod(110) = 17

1+2+4+10=17 = 111100

And, 205 x 71 mod(110) = 35

1+4+10+20=35 = 101110

After combining them we get the decoded text.
100100111100101110 which is our plain text.

UNIT-4

E-MAIL SECURITY

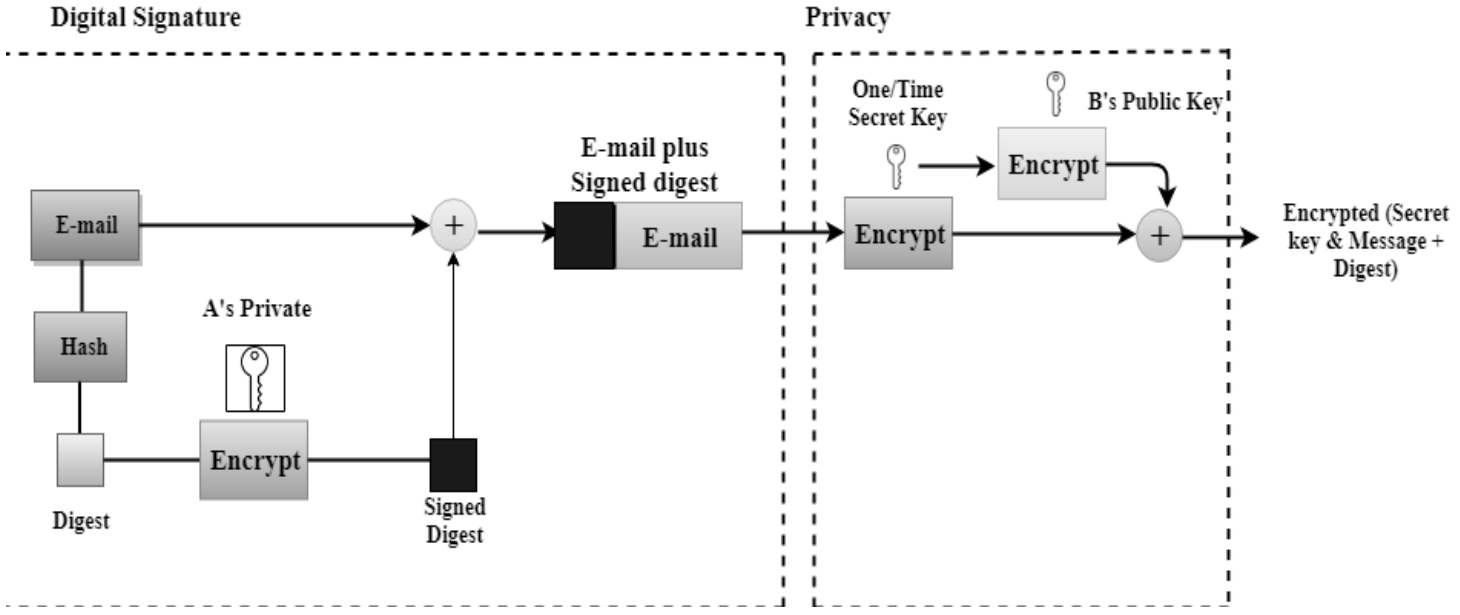
PGP (Pretty Good Privacy)

- PGP stands for Pretty Good Privacy (PGP) which is invented by Phil Zimmermann.
- PGP was designed to provide all four aspects of security, i.e., privacy, integrity, authentication, and non-repudiation in the sending of email.
- PGP uses a digital signature (a combination of hashing and public key encryption) to provide integrity, authentication, and non-repudiation. PGP uses a combination of secret key encryption and public key encryption to provide privacy. Therefore, we can say that the digital signature uses one hash function, one secret key, and two private-public key pairs.
- PGP is an open source and freely available software package for email security.
- PGP provides authentication through the use of Digital Signature.
- It provides confidentiality through the use of symmetric block encryption.
- It provides compression by using the ZIP algorithm, and EMAIL compatibility using the radix-64 encoding scheme.

Following are the steps taken by PGP to create secure e-mail at the sender site:

- The e-mail message is hashed by using a hashing function to create a digest.
- The digest is then encrypted to form a signed digest by using the sender's private key, and then signed digest is added to the original email message.
- The original message and signed digest are encrypted by using a one-time secret key created by the sender.
- The secret key is encrypted by using a receiver's public key.
- Both the encrypted secret key and the encrypted combination of message and digest are sent together.

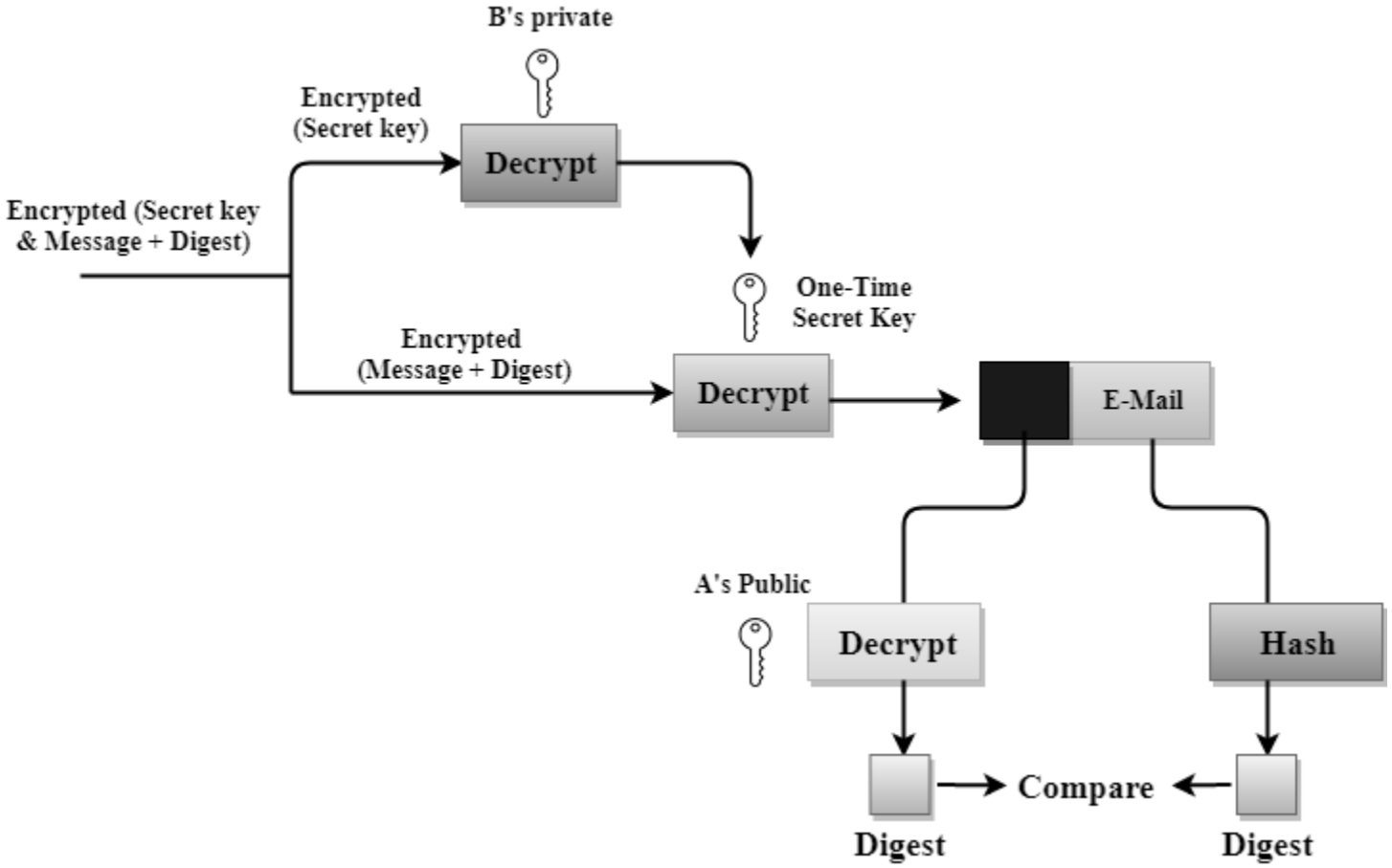
PGP at the Sender site (A)



Following are the steps taken to show how PGP uses hashing and a combination of three keys to generate the original message:

- The receiver receives the combination of encrypted secret key and message digest is received.
- The encrypted secret key is decrypted by using the receiver's private key to get the one-time secret key.
- The secret key is then used to decrypt the combination of message and digest.
- The digest is decrypted by using the sender's public key, and the original message is hashed by using a hash function to create a digest.
- Both the digests are compared if both of them are equal means that all the aspects of security are preserved.

PGP at the Receiver site (B)



Disadvantages of PGP Encryption

- **The Administration is difficult:** The different versions of PGP complicate the administration.
- **Compatibility issues:** Both the sender and the receiver must have compatible versions of PGP. For example, if you encrypt an email by using PGP with one of the encryption technique, the receiver has a different version of PGP which cannot read the data.
- **Complexity:** PGP is a complex technique. Other security schemes use symmetric encryption that uses one key or asymmetric encryption that uses two different keys. PGP

uses a hybrid approach that implements symmetric encryption with two keys. PGP is more complex, and it is less familiar than the traditional symmetric or asymmetric methods.

- **No Recovery:** Computer administrators face the problems of losing their passwords. In such situations, an administrator should use a special program to retrieve passwords. For example, a technician has physical access to a PC which can be used to retrieve a password. However, PGP does not offer such a special program for recovery; encryption methods are very strong so, it does not retrieve the forgotten passwords results in lost messages or lost files.

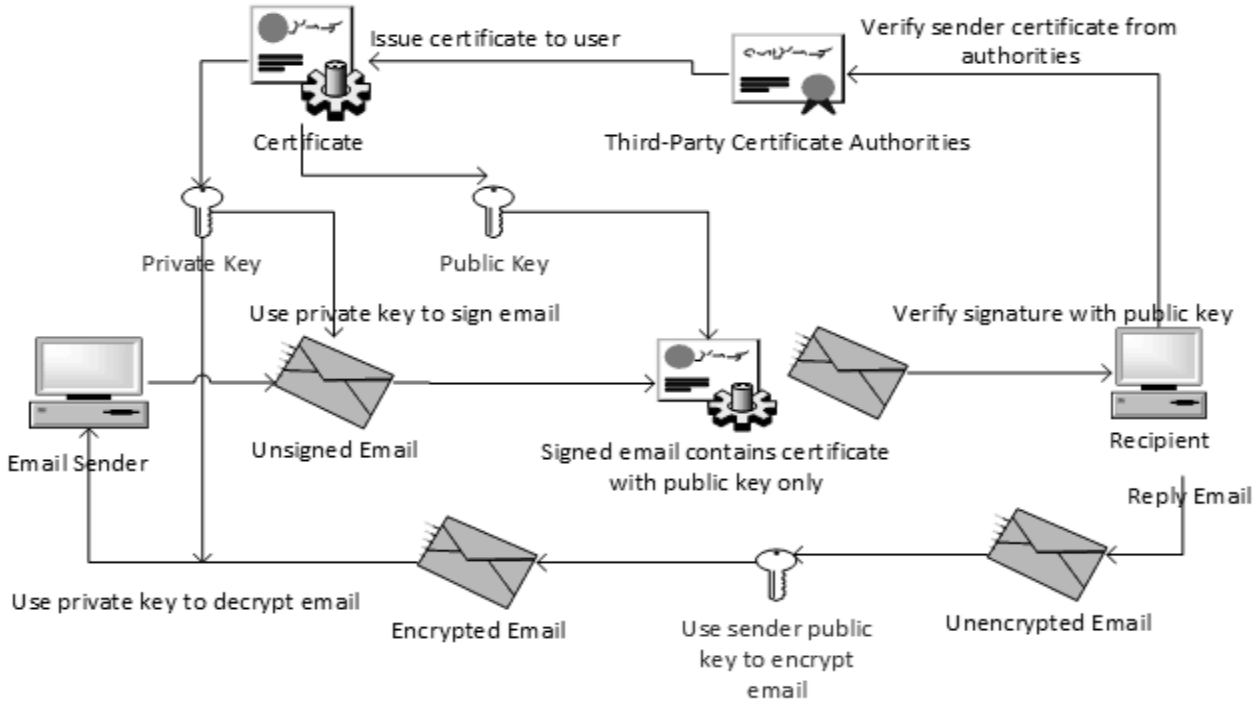
S/MIME

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, based on technology from RSA Data Security. S/MIME is defined in a number of documents, most importantly RFCs 3369, 3370, 3850 and 3851.

Multipurpose Internet Mail Extensions:-

MIME is an extension to the RFC 822 framework that is intended to address some of the problems and limitations of the use of SMTP (Simple Mail Transfer Protocol) or some other mail transfer protocol and RFC 822 for electronic mail. Following are the limitations of SMTP/822 scheme:

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



The five header fields defined in MIME are as follows:

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

Content-Transfer-Encoding: Indicates the type of transformation that has been used to represent the body of the message in a way that is acceptable for mail transport.

Content-ID: Used to identify MIME entities uniquely in multiple contexts.

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

MIME Content Types

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

S/MIME Functionality

In terms of general functionality, S/MIME is very similar to PGP. Both offer the ability to sign and/or encrypt messages. In this subsection, we briefly summarize S/MIME capability. We then look in more detail at this capability by examining message formats and message preparation.

Functions

S/MIME provides the following functions:

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

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

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

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

IP-SECURITY

IP security (IPSec) overview

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

Uses of IP Security –

IPsec can be used to do the following things:

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

Components of IP Security –

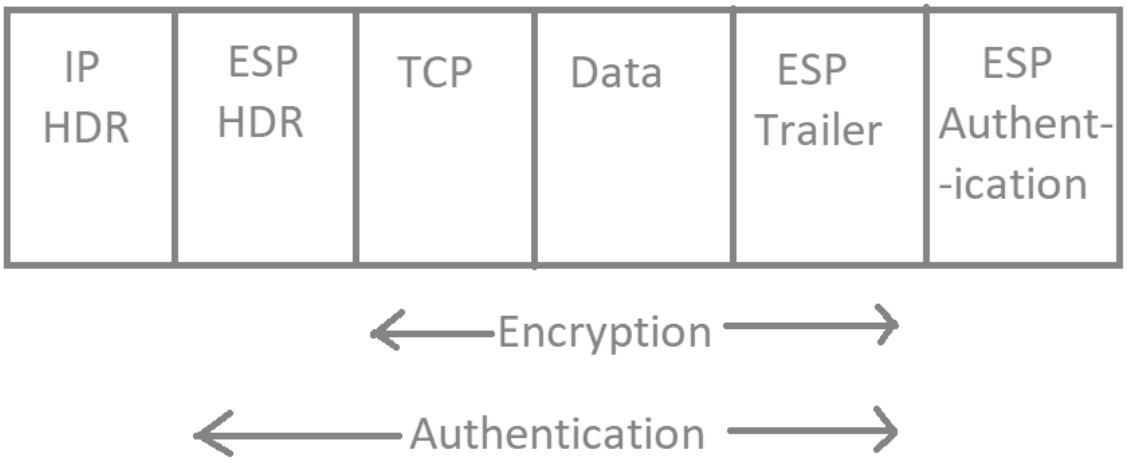
It has the following components:

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

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

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

Original Packet



Working of IP Security –

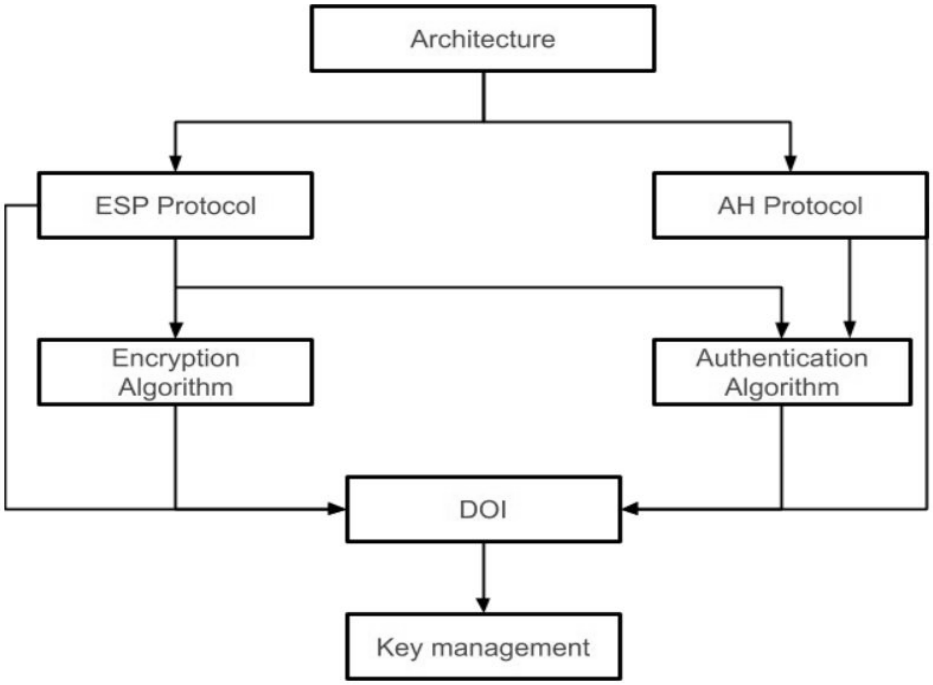
1. The host checks if the packet should be transmitted using IPsec or not. These packet traffic triggers the security policy for themselves. This is done when the system sending the packet apply an appropriate encryption. The incoming packets are also checked by the host that they are encrypted properly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

IPSec Architecture

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

- Confidentiality
- Authentication
- Integrity

IP Security Architecture:



1. Architecture:

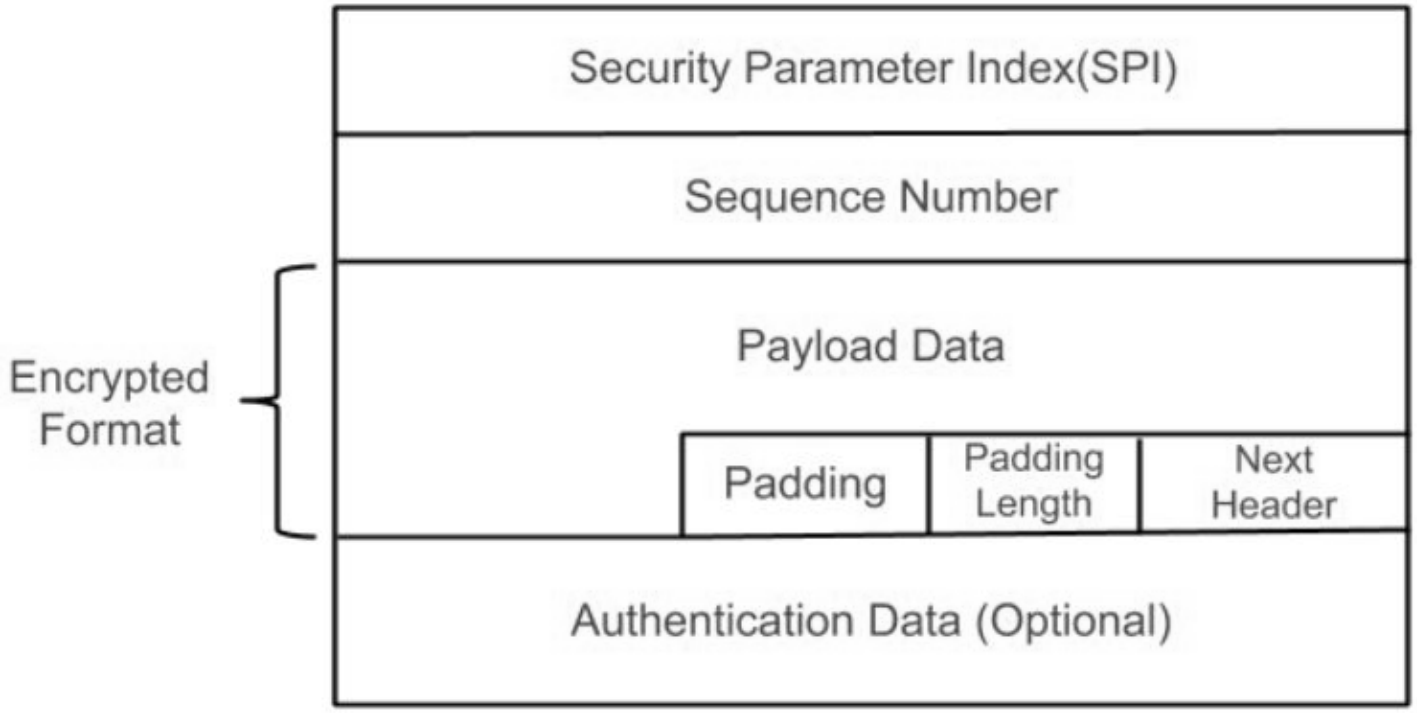
Architecture or IP Security Architecture covers the general concepts, definitions, protocols, algorithms and security requirements of IP Security technology.

2. ESP Protocol:

ESP(Encapsulation Security Payload) provide the confidentiality service. Encapsulation Security Payload is implemented in either two ways:

- ESP with optional Authentication.
- ESP with Authentication.

Packet Format:



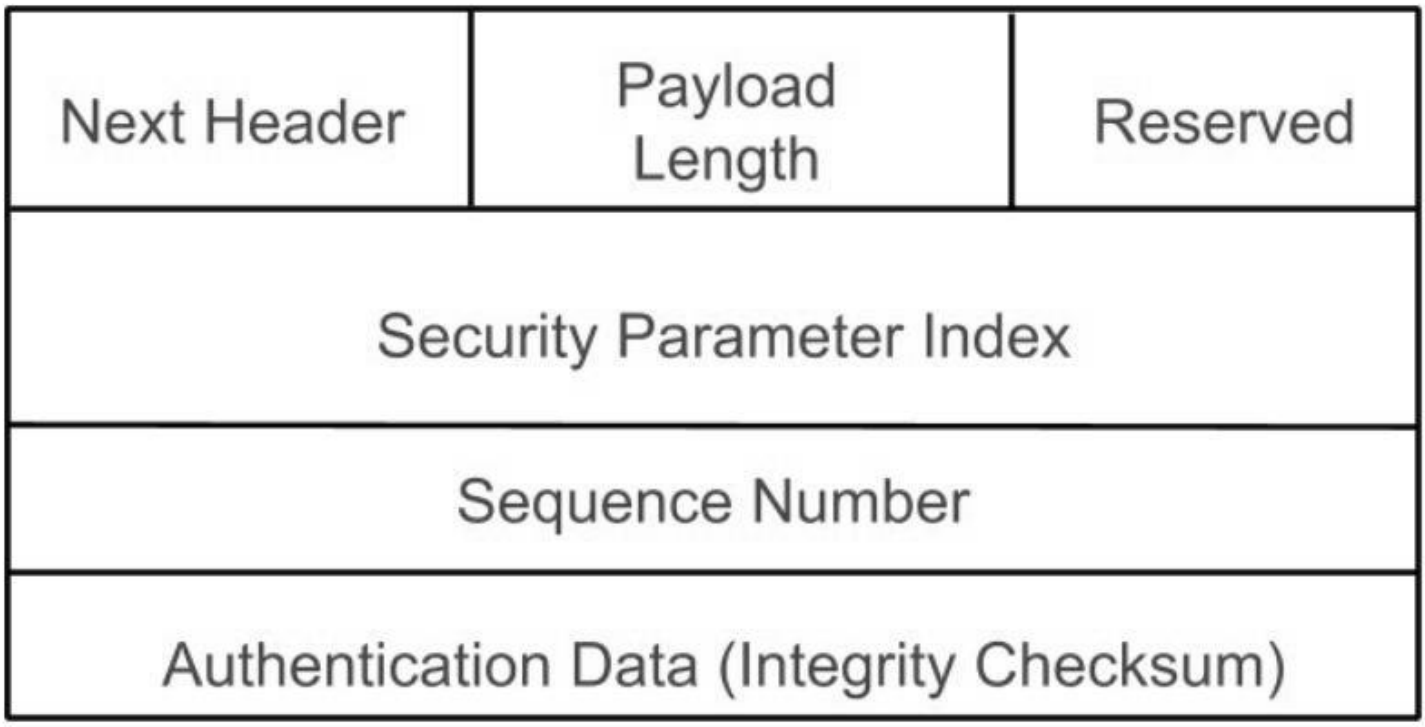
- **Security Parameter Index (SPI):**
  This parameter is used in Security Association. It is used to give a unique number to the connection build between Client and Server.
- **Sequence Number:**
  Unique Sequence number is allotted to every packet so that at the receiver side packets can be arranged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

3. Encryption algorithm:

Encryption algorithm is the document that describes various encryption algorithm used for Encapsulation Security Payload.

4. AH Protocol:

AH (Authentication Header) Protocol provides both Authentication and Integrity service. Authentication Header is implemented in one way only: Authentication along with Integrity. Authentication Header covers the packet format and general issue related to the use of AH for packet authentication and integrity.



5. Authentication Algorithm:

Authentication Algorithm contains the set of the documents that describe authentication algorithm used for AH and for the authentication option of ESP.

6. DOI (Domain of Interpretation):

DOI is the identifier which support both AH and ESP protocols. It contains values needed for documentation related to each other.

7. Key Management:

Key Management contains the document that describes how the keys are exchanged between sender and receiver.

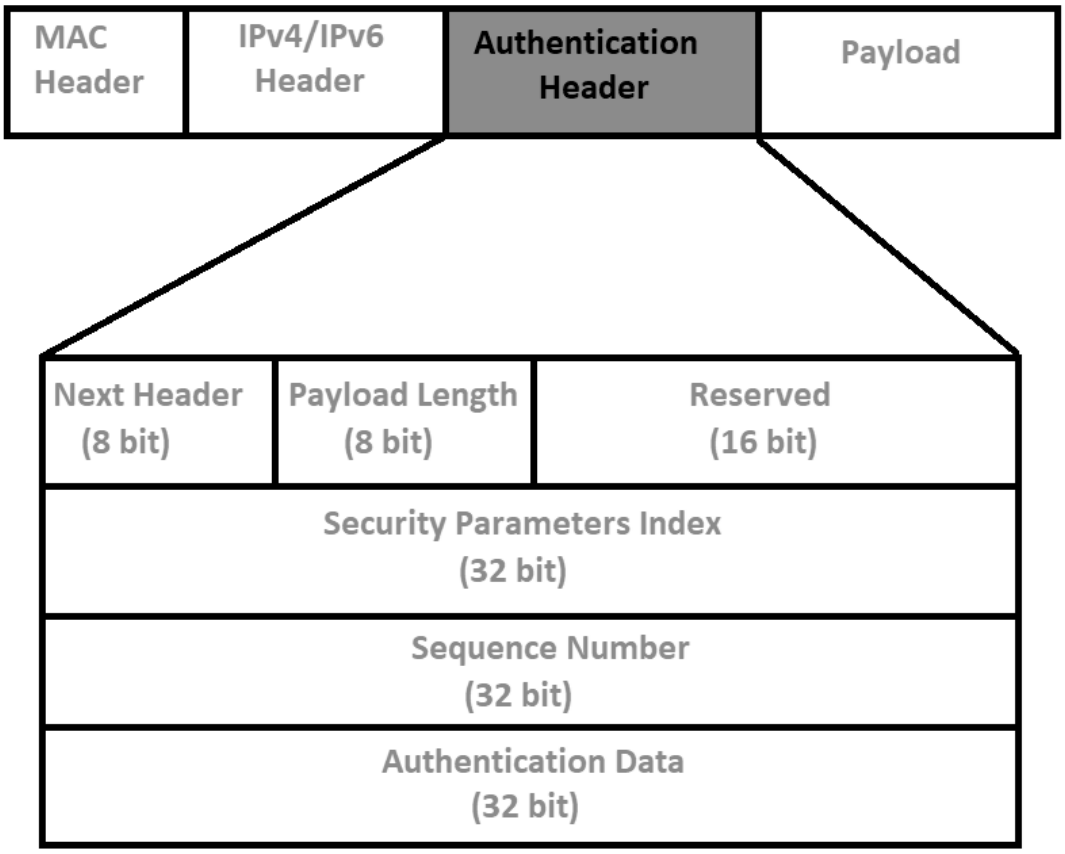
Authentication Header:

IP Authentication Header is used to provide connection-less integrity and data origin authentication. There are two main advantages that Authentication Header provides,

- **Message Integrity –**
  It means, message is not modified while coming from source.
- **Source Authentication –**
  It means, source is exactly source from which we were expecting data.
  When packet is sent from source A to Destination B, it consists of data that we need to send and header which consist of information regarding packet. Authentication Header verifies

origin of data and also payload to confirm if there has been modification done in between, during transmission between source and destination.

However, in transit, values of some IP header fields might change (like- Hop count, options, and extension headers). So, values of such fields cannot be protected from Authentication header. Authentication header cannot protect every field of IP header. It provides protection to fields which are essential to be protected.



1. **Next Header –**
   Next Header is 8-bit field that identifies type of header present after Authentication Header. In case of TCP, UDP or destination header or some other extension header it will store correspondence IP protocol number. Like, number 4 in this field will indicate IPv4, number 41 will indicate IPv6 and number 6 will indicate TCP.
2. **Payload Length –**
   Payload length is length of Authentication header and here we use scaling factor of 4. Whatever be size of header, divide it by 4 and then subtract by 2. We are subtracting by 2 because we're not counting first 8 bytes of Authentication header, which is first two row of picture given above. It means we are not including Next Header, Payload length, Reserved and Security Parameter index in calculating payload length. Like, say if payload length is given to be X. Then (X+2)*4 will be original Authentication header length.
3. **Reserved –**
   This is 16-bit field which is set to "zero" by sender as this field is reserved for future use.
4. **Security Parameter Index (SPI) –**
   It is arbitrary 32-bit field. It is very important field which identifies all packets which belongs to present connection. If we're sending data from Source A to Destination B. Both A and B will already know algorithm and key they are going to use. So for Authentication, hashing function and key will be required which only source and destination will know about. Secret

key between A and B is exchanged by method of Diffie Hellman algorithm. So Hashing algorithm and secret key for Security parameter index of connection will be fixed. Before data transfer starts security association needs to be established.

In **Security Association**, both parties needs to communicate prior to data exchange. Security association tells what is security parameter index, hashing algorithm and secret key that are being used.

5. **Sequence Number –**
   This unsigned 32-bit field contains counter value that increases by one for each packet sent. Every packet will need sequence number. It will start from 0 and will go till 232 – 1 and there will be no wrap around. Say, if all sequence numbers are over and none of it is left but we cannot wrap around as it is not allowed. So, we will end connection and re-establish connection again to resume transfer of remaining data from sequence number 0. Basically sequence numbers are used to stop replay attack.
6. **Authentication Data (Integrity Check Value) –**
   Authentication data is variable length field that contains Integrity Check Value (ICV) for packet. Using hashing algorithm and secret key, sender will create message digest which will be sent to receiver. Receiver on other hand will use same hashing algorithm and secret key. If both messages digest matches then receiver will accept data. Otherwise, receiver will discard it by saying that message has been modified in between. So basically, authentication data is used to verify integrity of transmission. Also length of Authentication data depends upon hashing algorithm we choose.

Encapsulating Security Payload (ESP)

- Encapsulating Security Payload (ESP) provides all encryption services in IPSec based on integrity for the payload and not for the IP header, confidentiality and authentication that using encryption, without authentication is strongly discouraged because it is insecure.
- Any translations in readable message format into an unreadable format are encrypted and used to hide the message content against data tampering.
- IPSec provides an open framework, such as SHA and MD5 for implementing industry standard algorithms.
- Encryption/decryption allows only the sender and the authorised receiver to make the data to be received in readable form and only after the integrity verification process is complete, the data payload in the packet is decrypted.
- IPSec uses a unique identifier for each packet, which is a data equivalent of a fingerprint and checks for packets that are authorized or not. It doesn't sign the entire packet unless it is being tunneled—ordinarily, for this IP data payload is protected, not the IP header. In Tunnel Mode, where the entire original IP packet is encapsulated with a new packet header added.
- ESP in transport mode does not provide integrity and authentication for the entire IP packet.

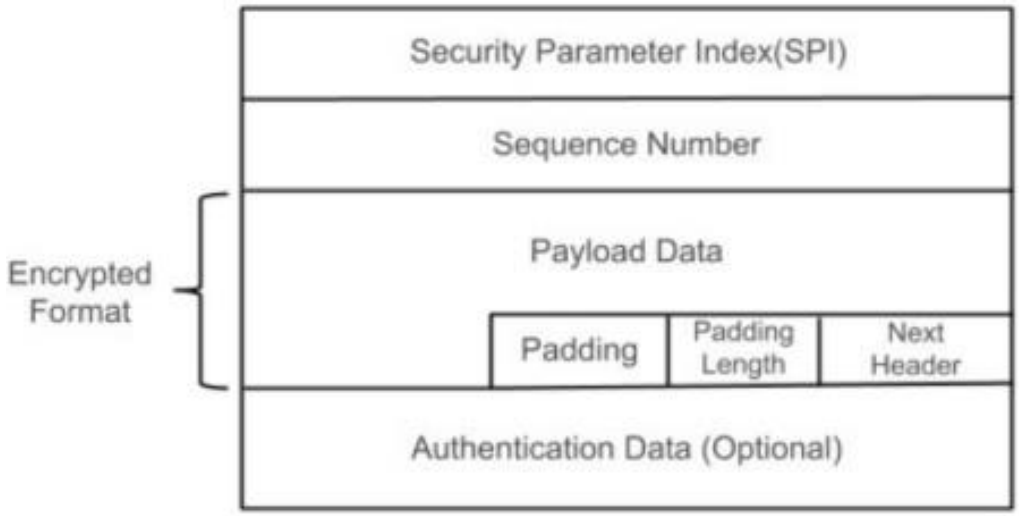
ESP Format

The ESP format is diagrammatically represented as follows –

Security Parameters Index (32 bits) – Identifies a security association. This field is mandatory. The value of zero is reserved for local, implementation- specific use and MUST NOT be sent on the wire.

Sequence Number (32 bits) – A monotonically increasing counter value; this provides an anti-replay function, as discussed for AH. The first packet sent using a given SA will have a Sequence number of 1.

Payload Data (variable) – this is a transport-level segment (transport mode) or IP packet (tunnel mode) that is protected by encryption. The type of content that was protected is indicated by the Next Header field.



Padding (0-255 bytes) – Padding for encryption, to extend the payload data to a size that fits the encryption's cipher block size, and to align the next field.

Pad Length (8 bits) – Indicates the number of pad bytes immediately preceding this field.

Next Header (8 bits) – Identifies the type of data contained in the payload data field by identifying the first header in that payload.

Authentication Data (variable) – A variable-length field (must be an integral number of 32-bit words) that contains the Integrity. Check Value computed over the ESP packet minus the Authentication Data field. This field is optional and is included only if the authentication service has been selected for the SA in question.

COMBINING SECURITY ASSOCIATIONS

An individual SA can implement either the AH or ESP protocol but not both. Sometimes a particular traffic flow will call for the services provided by both AH and ESP. Further, a particular traffic flow may require IPSec services between hosts and, for that same flow, separate services between security gateways, such as firewalls. In all of these cases, multiple SAs must be employed for the same traffic flow to achieve the desired IPSec services. The term *security association bundle* refers to a sequence of SAs through which traffic must be processed to provide a desired set of IPSec services. The SAs in a bundle may terminate at different endpoints or at the same endpoints.

Security associations may be combined into bundles in two ways:

- **Transport adjacency:** Refers to applying more than one security protocol to the same IP packet, without invoking tunneling. This approach to combining AH and ESP allows for only one level of combination; further nesting yields no added benefit since the processing is performed at one IPsec instance: the (ultimate) destination.
- **Iterated tunneling:** Refers to the application of multiple layers of security protocols affected through IP tunneling. This approach allows for multiple levels of nesting, since each tunnel can

originate or terminate at a different IPsec site along the path. The two approaches can be combined, for example, by having a transport SA between hosts travel part of the way through a tunnel SA between security gateways.

Authentication plus Confidentiality: Encryption and authentication can be combined in order to transmit an IP packet that has both confidentiality and authentication between hosts. We look at several approaches.

ESP with Authentication Option In this approach, the user first applies ESP to the data to be protected and then appends the authentication data field. There are actually two sub cases:

- **Transport mode ESP:** Authentication and encryption apply to the IP payload delivered to the host, but the IP header is not protected.
- **Tunnel mode ESP:** Authentication applies to the entire IP packet delivered to the outer IP destination address (e.g., a firewall), and authentication is performed at that destination. The entire inner IP packet is protected by the privacy mechanism, for delivery to the inner IP destination. For both cases, authentication applies to the cipher text rather than the plaintext.

Transport Adjacency: Another way to apply authentication after encryption is to use two bundled transport SAs, with the inner being an ESP SA and the outer being an AH SA. In this case ESP is used without its authentication option. Because the inner SA is a transport SA, encryption is applied to the IP payload. The resulting packet consists of an IP header (and possibly IPv6 header extensions) followed by an ESP. AH is then applied in transport mode, so that authentication covers the ESP plus the original IP header (and extensions) except for mutable fields.

The advantage of this approach over simply using a single ESP SA with the ESP authentication option is that the authentication covers more fields, including the source and destination IP addresses. The disadvantage is the overhead of two SAs versus one SA.

Transport-Tunnel Bundle: The use of authentication prior to encryption might be preferable for several reasons. First, because the authentication data are protected by encryption, it is impossible for anyone to intercept the message and alter the authentication data without detection. Second, it may be desirable to store the authentication information with the message at the destination for later reference.

IP Key Management

- A protocol and cryptographic technique
- Application layer protocol for IPSP
- Independent of IPSP
- Initially supporting public key techniques
- Later adding Key Distribution Center (e.g., Kerbos) and/or manual

Requirements of IKMP(IP Key Management Protocol)

The basic functions of the IKMP are to:

- create,
- manage, and
- remove security associations

Used by the IP Security Protocol (IPSP) or other similar security protocols.
A security association consists of:

- the key(s) used for that association as well as
- attributes guiding the operation of the associated protocol.

In particular, the IKMP is expected to handle negotiation of:

- cryptographic algorithms,
- protocol format, and
- protocol options (e.g., security labels, integrity checks).

IKMP Functional Requirements

Functional requirements for IKMP include:

- Security Association ID (SAID) assignment (or Security Association-Creation)
- Key generation/distribution
- Attribute Negotiation
- Terminate/Delete Association
- Security Association Maintenance
- Peer Discovery and Authentication
- Recovery
- Protocol Profile
- Multiparty Associations Key Management

IKMP Issues

- Device name and address implications for directories and certificates
- Can a SA change, or is a change accomplished by terminating an old SA and establishing a new one??
- Shared keys - used for multicast or (possibly) multiple IPSP routers serving a site
- Relationship to other IETF Key Management related activities

Web Security

Web Security Considerations

Web Security is very important nowadays. Websites are always prone to security threats/risks. Web Security deals with the security of data over the internet/network or web or while it is being transferred to the internet. For e.g. when you are transferring data between client and server and you have to protect that data that security of data is your web security.

Hacking of Website may result in theft of Important Customer Data the important customer data maybe your credit card information the login details of the customer or it can be the destruction of one's business and propagation of illegal content to the users so while somebody hacks your website they can either steal the information the important information of the customers or they can even propagate the illegal content to your users through your website so, therefore, security considerations are needed in the context of web security.

Security Threats:

A Threat is nothing but a possible event that can damage and harm an information system. Security Threat is defined as a risk that which, can potentially harm Computer systems & organizations. Whenever an Individual or an Organization creates a website, they are vulnerable to security attacks.

Security attacks are mainly aimed at stealing altering or destroying a piece of personal and confidential information, stealing the hard drive space, illegally accessing passwords. So whenever the website you created is vulnerable to security attacks then the attacks are going

to steal your data alter your data destroy your personal information see your confidential information and also it accessing your password.

Top Web Security Threats:

Web security threats are constantly emerging and evolving, but many threats consistently appear at the top of the list of web security threats. These include:

- Cross-site scripting (XSS)
- SQL Injection
- Phishing
- Ransomware
- Code Injection
- Viruses and worms
- Spyware
- Denial of Service

Security Consideration:

- **Updated Software:** You need to always update your software. Hackers may be aware of vulnerabilities in certain software, which are sometimes caused by bugs and can be used to damage your computer system and steal personal data. Older versions of software can become a gateway for hackers to enter your network. Software makers soon become aware of these vulnerabilities and will fix vulnerable or exposed areas. That's why It is mandatory to keep your software updated, It plays an important role in keeping your personal data secure.
- **Beware of SQL Injection:** SQL Injection is an attempt to manipulate your data or your database by inserting a rough code into your query. For e.g. somebody can send a query to your website and this query can be a rough code while it gets executed it can be used to manipulate your database such as change tables, modify or delete data or it can retrieve important information also so, one should be aware of the SQL injection attack.
- **Cross-Site Scripting (XSS):** XSS allows the attackers to insert client-side script into web pages. E.g. Submission of forms. It is a term used to describe a class of attacks that allow an attacker to inject client-side scripts into other users' browsers through a website. As the injected code enters the browser from the site, the code is reliable and can do things like sending the user's site authorization cookie to the attacker.
- **Error Messages:** You need to be very careful about error messages which are generated to give the information to the users while users access the website and some error messages are generated due to one or another reason and you should be very careful while providing the information to the users. For e.g. login attempt – If the user fails to login the error message should not let the user know which field is incorrect: Username or Password.
- **Data Validation:** Data validation is the proper testing of any input supplied by the user or application. It prevents improperly created data from entering the information system. Validation of data should be performed on both server-side and client-side. If we perform data validation on both sides that will give us the authentication. Data validation should occur when data is received from an outside party, especially if the data is from untrusted sources.
- **Password:** Password provides the first line of defense against unauthorized access to your device and personal information. It is necessary to use a strong password. Hackers in many cases use sophisticated software that uses brute force to crack passwords. Passwords must be complex to protect against brute force. It is good to enforce password requirements such as a minimum of eight characters long must including uppercase letters, lowercase letters, special characters, and numerals.

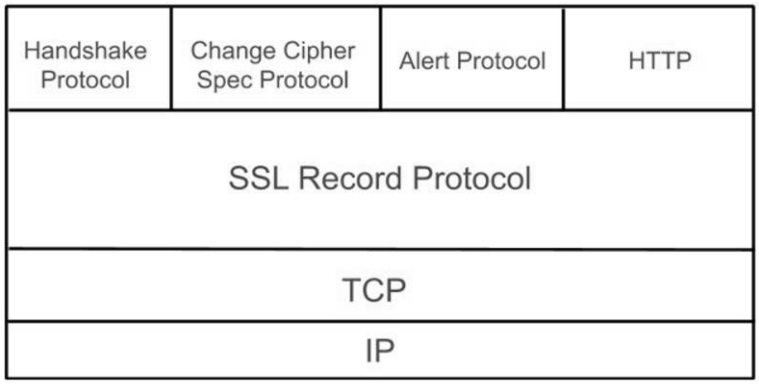
Secure Socket Layer (SSL)

Secure Socket Layer (SSL) provides security to the data that is transferred between web browser and server. SSL encrypts the link between a web server and a browser which ensures that all data passed between them remain private and free from attack.

Secure Socket Layer Protocols:

- SSL record protocol
- Handshake protocol
- Change-cipher spec protocol
- Alert protocol

SSL Protocol Stack:

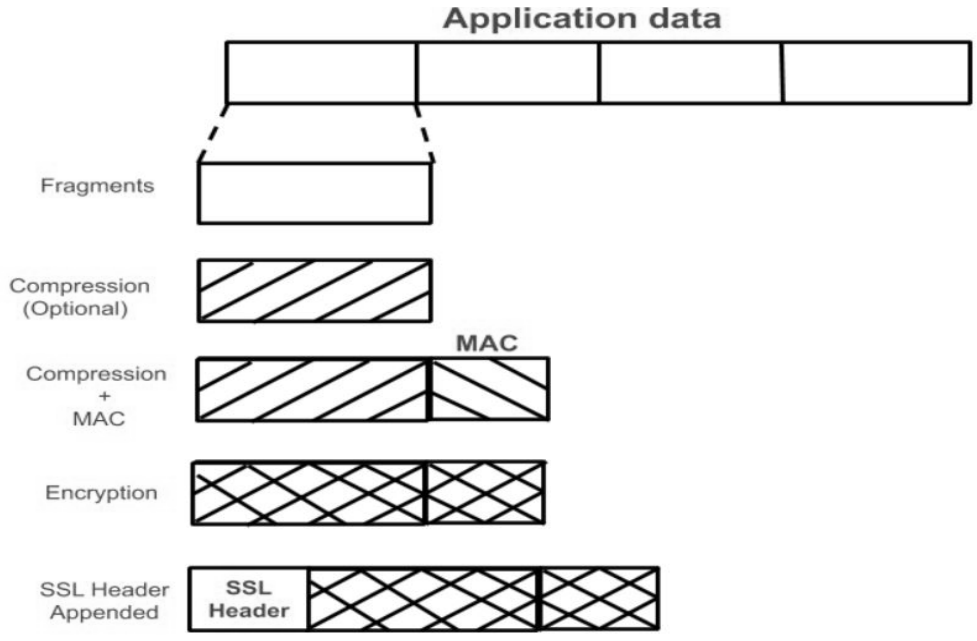


SSL Record Protocol:

SSL Record provides two services to SSL connection.

- Confidentiality
- Message Integrity

In the SSL Record Protocol application data is divided into fragments. The fragment is compressed and then encrypted MAC (Message Authentication Code) generated by algorithms like SHA (Secure Hash Protocol) and MD5 (Message Digest) is appended. After that encryption of the data is done and in last SSL header is appended to the data.



Handshake Protocol:

Handshake Protocol is used to establish sessions. This protocol allows the client and server to authenticate each other by sending a series of messages to each other. Handshake protocol uses four phases to complete its cycle.

- **Phase-1:** In Phase-1 both Client and Server send hello-packets to each other. In this IP session, cipher suite and protocol version are exchanged for security purposes.
- **Phase-2:** Server sends his certificate and Server-key-exchange. The server end phase-2 by sending the Server-hello-end packet.
- **Phase-3:** In this phase Client reply to the server by sending his certificate and Client-exchange-key.
- **Phase-4:** In Phase-4 Change-cipher suite occurred and after this Handshake Protocol ends.

Change-cipher Protocol:

This protocol uses the SSL record protocol. Unless Handshake Protocol is completed, the SSL record Output will be in a pending state. After handshake protocol, the Pending state is converted into the current state.

Change-cipher protocol consists of a single message which is 1 byte in length and can have only one value. This protocol's purpose is to cause the pending state to be copied into the current state.

Alert Protocol:

This protocol is used to convey SSL-related alerts to the peer entity. Each message in this protocol contains 2 bytes.

The level is further classified into two parts:

- **Warning:**
  This Alert has no impact on the connection between sender and receiver.

- **Fatal Error:**
  This Alert breaks the connection between sender and receiver.

Silent Features of Secure Socket Layer:

- The advantage of this approach is that the service can be tailored to the specific needs of the given application.
- Secure Socket Layer was originated by Netscape.
- SSL is designed to make use of TCP to provide reliable end-to-end secure service.
- This is a two-layered protocol.

Transport Layer Security (TLS)

Transport Layer Securities (TLS) are designed to provide security at the transport layer. TLS was derived from a security protocol called Secure Socket Layer (SSL). TLS ensures that no third party may eavesdrop or tampers with any message.
There are several benefits of TLS:

- **Encryption:**
  TLS/SSL can help to secure transmitted data using encryption.
- **Interoperability:**
  TLS/SSL works with most web browsers, including Microsoft Internet Explorer and on most operating systems and web servers.
- **Algorithm flexibility:**
  TLS/SSL provides operations for authentication mechanism, encryption algorithms and hashing algorithm that are used during the secure session.
- **Ease of Deployment:**
  Many applications TLS/SSL temporarily on a windows server 2003 operating systems.
- **Ease of Use:**
  Because we implement TLS/SSL beneath the application layer, most of its operations are completely invisible to client.

Working of TLS:
The client connect to server (using TCP), the client will be something. The client sends number of specification:
1. Version of SSL/TLS.
2. Which cipher suites, compression method it wants to use.

The server checks what the highest SSL/TLS version is that is supported by them both, picks a cipher suite from one of the clients option (if it supports one) and optionally picks a compression method. After this the basic setup is done, the server provides its certificate. This certificate must be trusted either by the client itself or a party that the client trusts. Having verified the certificate and being certain this server really is who he claims to be (and not a man in the middle), a key is exchanged. This can be a public key, "PreMasterSecret" or simply nothing depending upon cipher suite.

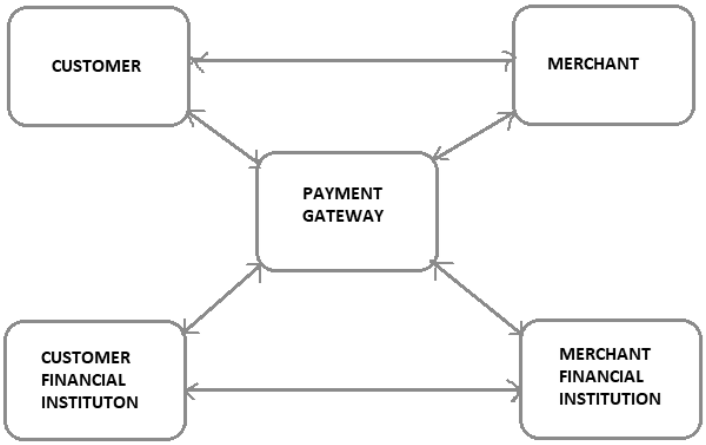
Both the server and client can now compute the key for symmetric encryption. The handshake is finished and the two hosts can communicate securely. To close a connection by finishing. TCP connection both sides will know the connection was improperly terminated. The connection cannot be compromised by this through, merely interrupted.

Secure Electronic Transaction (SET) Protocol

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

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

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



Requirements in SET:

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

- It has to provide mutual authentication i.e., customer (or cardholder) authentication by confirming if the customer is an intended user or not, and merchant authentication.
- It has to keep the PI (Payment Information) and OI (Order Information) confidential by appropriate encryptions.
- It has to be resistive against message modifications i.e., no changes should be allowed in the content being transmitted.
- SET also needs to provide interoperability and make use of the best security mechanisms.

Participants in SET:

In the general scenario of online transactions, SET includes similar participants:

1. **Cardholder –** customer
2. **Issuer –** customer financial institution
3. **Merchant**
4. **Acquirer –** Merchant financial
5. **Certificate authority –** Authority that follows certain standards and issues certificates(like X.509V3) to all other participants.

SET functionalities:

- **Provide Authentication**
    - **Merchant Authentication** – To prevent theft, SET allows customers to check previous relationships between merchants and financial institutions. Standard X.509V3 certificates are used for this verification.
    - **Customer / Cardholder Authentication** – SET checks if the use of a credit card is done by an authorized user or not using X.509V3 certificates.

- **Provide Message Confidentiality**: Confidentiality refers to preventing unintended people from reading the message being transferred. SET implements confidentiality by using encryption techniques. Traditionally DES is used for encryption purposes.
- **Provide Message Integrity**: SET doesn't allow message modification with the help of signatures. Messages are protected against unauthorized modification using RSA digital signatures with SHA-1 and some using HMAC with SHA-1.

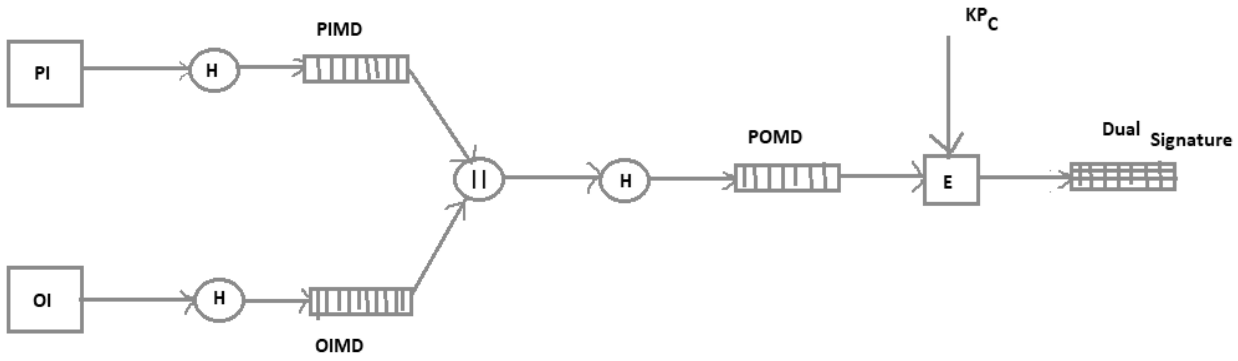
Dual Signature :

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

Order Information (OI) for merchant

Payment Information (PI) for bank

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



Where,

PI stands for payment information

OI stands for order information

PIMD stands for Payment Information Message Digest

OIMD stands for Order Information Message Digest

POMD stands for Payment Order Message Digest

H stands for Hashing

E stands for public key encryption

KPc is customer's private key

|| stands for append operation

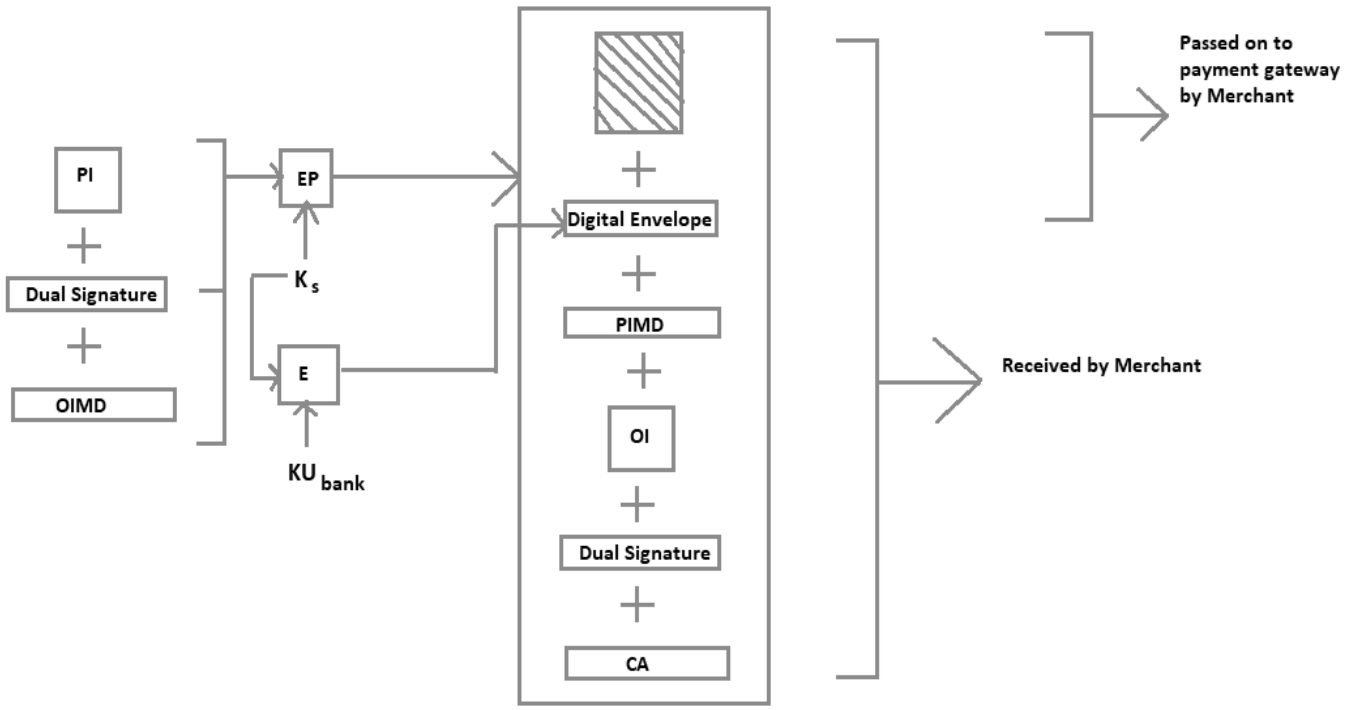
Dual signature, DS= E(KPc, [H(H(PI)||H(OI))])

Purchase Request Generation:

The process of purchase request generation requires three inputs:

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

The purchase request is generated as follows:



Here,

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

The new things are :

EP which is symmetric key encryption

Ks is a temporary symmetric key

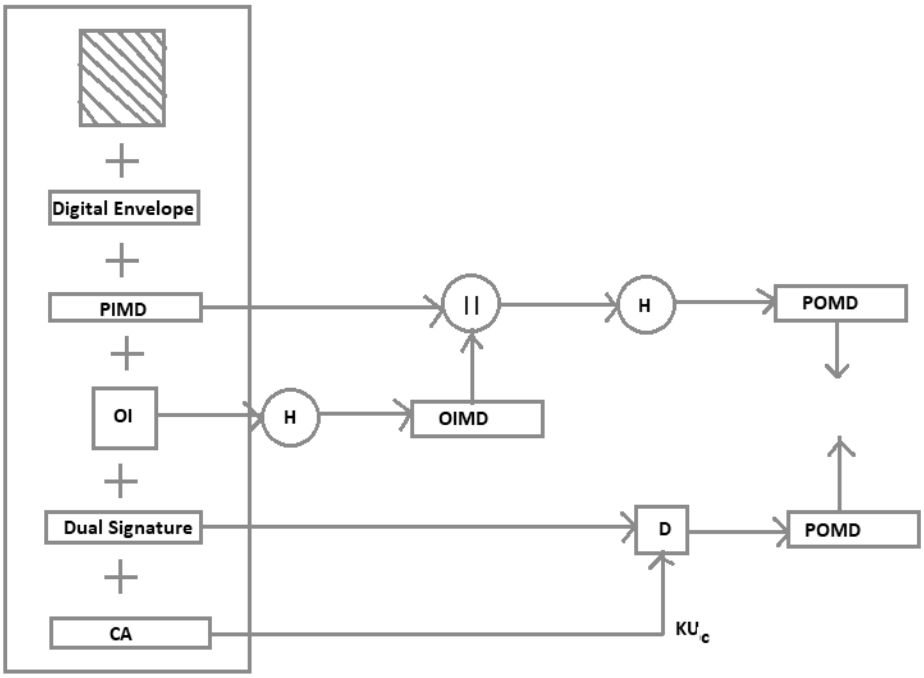
KUbank is public key of bank

CA is Cardholder or customer Certificate

Digital Envelope = E(KUbank, Ks)

Purchase Request Validation on Merchant Side:

The Merchant verifies by comparing POMD generated through PIMD hashing with POMD generated through decryption of Dual Signature as follows:



Since we used Customer's private key in encryption here we use KUC which is the public key of the customer or cardholder for decryption 'D'.

Payment Authorization and Payment Capture:

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

**********THE END**********