

INTRODUCTION: Computer data often travels from one computer to another, leaving the safety of its protected physical surroundings. Once the data is out of hand, people with bad intention could modify or forge your data, either for amusement or for their own benefit. Cryptography can reformat and transform our data, making it safer on its trip between computers. The technology is based on the essentials of secret codes, augmented by modern mathematics that protects our data in powerful ways.

- **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

THE OSI SECURITY ARCHITECTURE

To assess effectively the security needs of an organization and to evaluate and choose various security products and policies, the manager responsible for security needs some systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. The OSI security architecture was developed in the context of the OSI protocol architecture, which is described in Appendix H. However, for our purposes in this chapter, an understanding of the OSI protocol architecture is not required. For our purposes, the OSI security architecture provides a useful, if abstract, overview of many of the concepts.. The OSI security architecture focuses on security attacks, mechanisms, and services. These can be defined briefly as follows:

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

- **Security attack** – Any action that compromises the security of information owned by an organization.
- **Security mechanism** – A mechanism that is designed to detect, prevent or recover from a security attack.
- **Security service** – A service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks and they make use of one or more security mechanisms to provide the service.

Table 1.1. Threats and Attacks (RFC 2828)

Threat

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

Attack

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

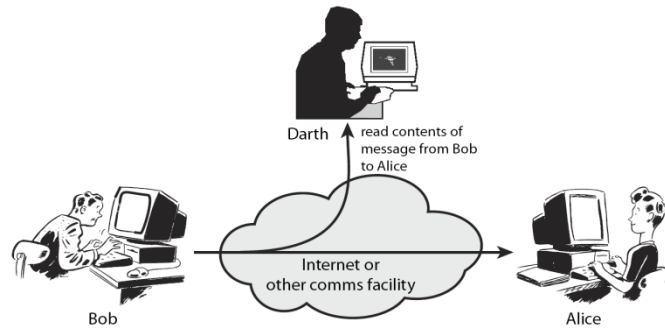
Security Attacks:

A useful means of classifying security attacks used both in X.800 and RFC 2828, is in terms of passive attacks and active attacks. It 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.

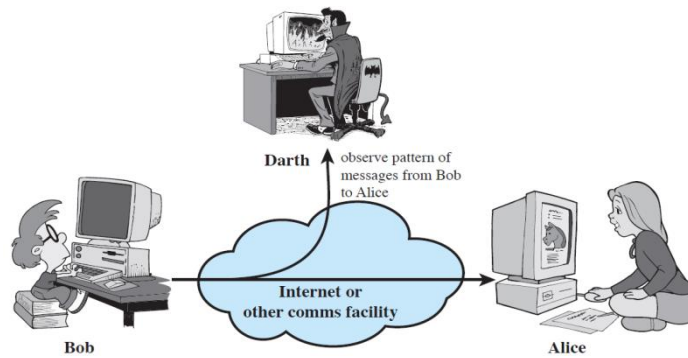
Passive Attacks:

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 the release of message contents and traffic analysis. 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.



Release of Message Contents

A second type of passive attack, traffic analysis, is subtler. 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.



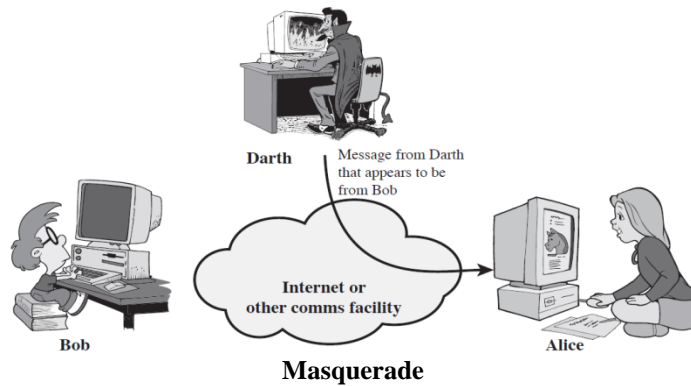
Traffic Analysis

Passive attacks are very difficult to detect, because they do not involve any alteration of the data. Typically, the message traffic is sent and received in an apparently normal fashion, and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern. However, it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

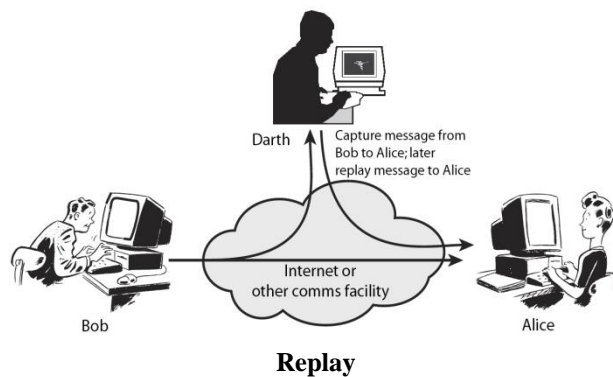
Active Attacks

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

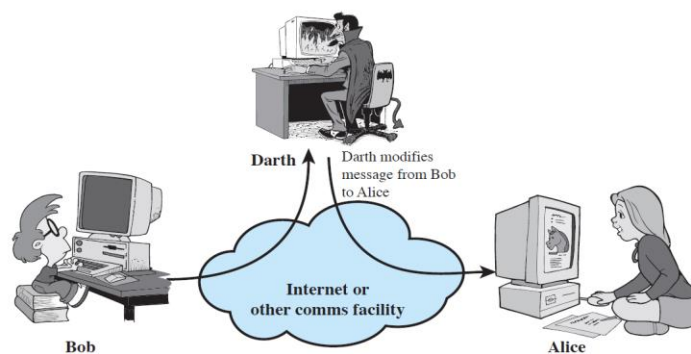
A Masquerade takes place when one entity pretends to be a different entity. A masquerade attack usually includes one of the other forms of active attack.



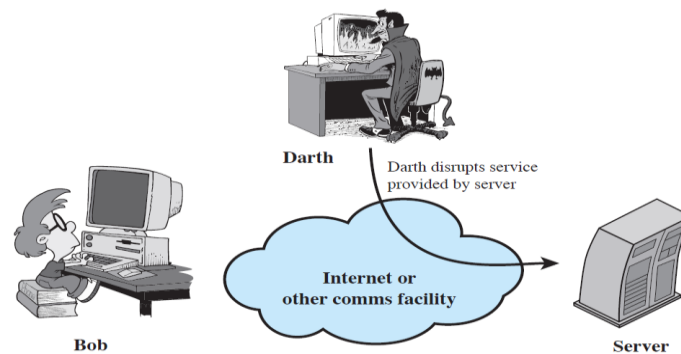
Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.



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”.



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.



Denial of Service

SECURITY SERVICES:

A processing or communication service that is provided by a system to give a specific kind of protection to system resources; security services implement security policies and are implemented by security mechanisms.

X.800 divides these services into five categories and fourteen specific services

AUTHENTICATION	DATA INTEGRITY
<p>The assurance that the communicating entity is the one that it claims to be.</p> <p>Peer Entity Authentication Used in association with a logical connection to provide confidence in the identity of the entities connected.</p> <p>Data-Origin Authentication In a connectionless transfer, provides assurance that the source of received data is as claimed.</p>	<p>The assurance that data received are exactly as sent by an authorized entity (i.e., contain no modification, insertion, deletion, or replay).</p> <p>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.</p>
ACCESS CONTROL	Connection Integrity without Recovery
<p>The prevention of unauthorized use of a resource (i.e., this service controls who can have access to a resource, under what conditions access can occur, and what those accessing the resource are allowed to do).</p>	<p>As above, but provides only detection without recovery.</p>
DATA CONFIDENTIALITY	Selective-Field Connection Integrity
<p>The protection of data from unauthorized disclosure.</p> <p>Connection Confidentiality The protection of all user data on a connection.</p> <p>Connectionless Confidentiality The protection of all user data in a single data block</p>	<p>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.</p> <p>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.</p>

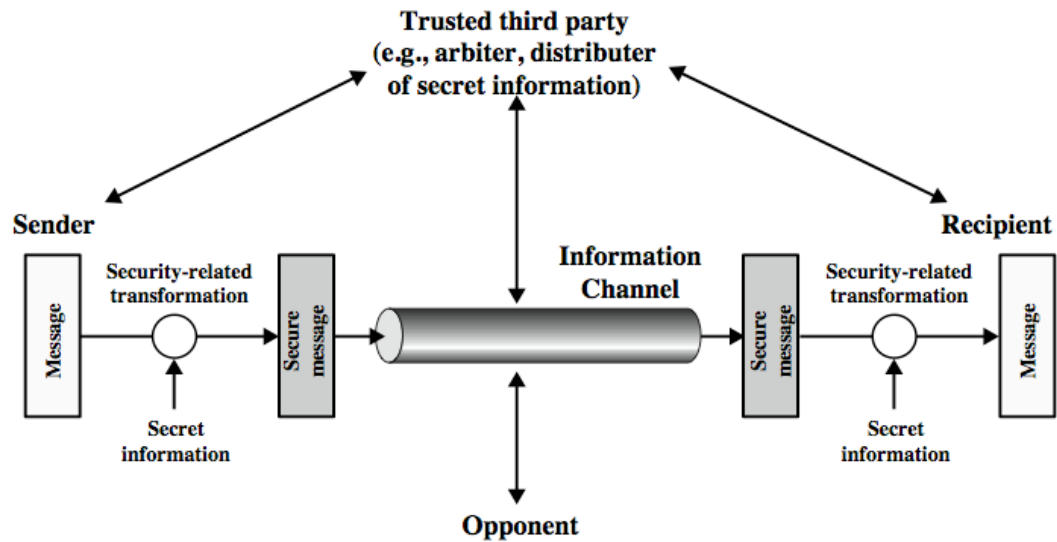
<p>Selective-Field Confidentiality The confidentiality of selected fields within the user data on a connection or in a single data block.</p> <p>Traffic-Flow Confidentiality The protection of the information that might be derived from observation of traffic flows.</p>	<p>Selective-Field Connectionless 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.</p> <p>NONREPUDIATION Provides protection against denial by one of the entities involved in a communication of having participated in all or part of the communication.</p> <p>Nonrepudiation, Origin Proof that the message was sent by the specified party.</p> <p>Nonrepudiation, Destination Proof that the message was received by the specified party</p>
--	---

Security Mechanisms:

The mechanisms are divided into those that are implemented in a specific protocol layer, such as TCP or an application-layer protocol, and those that are not specific to any particular protocol layer or security service.

SPECIFIC SECURITY MECHANISMS	PERVASIVE SECURITY MECHANISMS
<p>May be incorporated into the appropriate protocol layer in order to provide some of the OSI security services.</p> <p>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.</p> <p>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).</p> <p>Access Control A variety of mechanisms that enforce access rights to resources.</p> <p>Data Integrity A variety of mechanisms used to assure the integrity of a data unit or stream of data units.</p> <p>Authentication Exchange A mechanism intended to ensure the identity of an entity by means of information exchange.</p> <p>Traffic Padding The insertion of bits into gaps in a data stream to frustrate traffic analysis attempts.</p> <p>Routing Control Enables selection of particular physically secure routes for certain data and allows routing changes, especially when a breach of security is suspected.</p> <p>Notarization The use of a trusted third party to assure certain properties of a data exchange.</p>	<p>Mechanisms that are not specific to any particular OSI security service or protocol layer.</p> <p>Trusted Functionality That which is perceived to be correct with respect to some criteria (e.g., as established by a security policy).</p> <p>Security Label The marking bound to a resource (which may be a data unit) that names or designates the security attributes of that resource.</p> <p>Event Detection Detection of security-relevant events.</p> <p>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.</p> <p>Security Recovery Deals with requests from mechanisms, such as event handling and management functions, and takes recovery actions.</p>

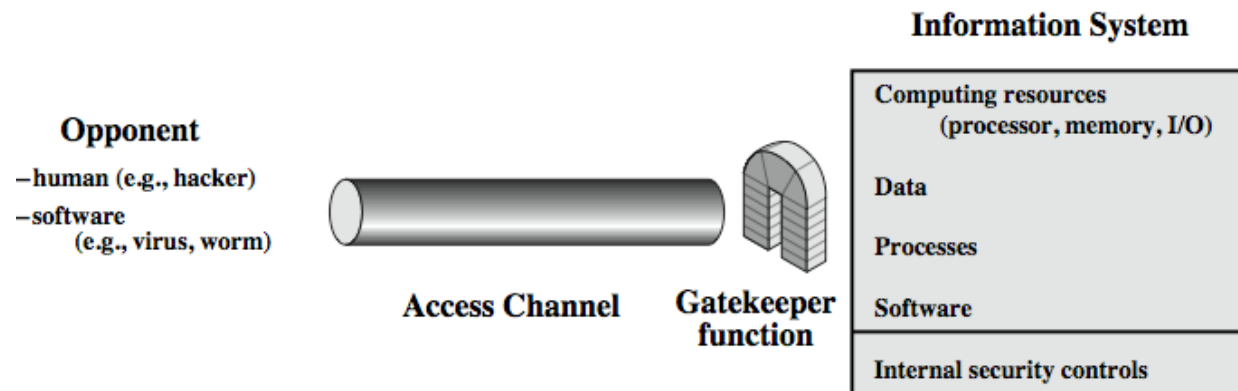
A MODEL FOR NETWORK SECURITY



A message is to be transferred from one party to another across some sort of internet. The two parties, who are the principals in this transaction, must cooperate for the exchange to take place. A logical information channel is established 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. **using this model requires us to:**

- design a suitable algorithm for the security transformation
- generate the secret information (keys) used by the algorithm
- develop methods to distribute and share the secret information
- specify a protocol enabling the principals to use the transformation and secret information for a security service

MODEL FOR NETWORK ACCESS SECURITY



Information Security unit-I

- **using this model requires us to:**
 - select appropriate gatekeeper functions to identify users
 - implement security controls to ensure only authorized users access designated information or resources
- **trusted computer systems can be used to implement this model**

CONVENTIONAL ENCRYPTION

Symmetric encryption also referred to as conventional encryption or single-key encryption was the only type of encryption in use prior to the development of public key encryption in the 1970s.

Some basic terminologies used:

- **plaintext** - the original message
- **ciphertext** - the coded message
- **encipher (encrypt)** - converting plaintext to ciphertext
- **decipher (decrypt)** - restoring the plaintext from the ciphertext
- **cryptography** - study of encryption principles/methods
- **cryptanalysis (codebreaking)** - the study of principles/ methods of deciphering ciphertext *without* knowing key
- **cryptology** - the field of both cryptography and cryptanalysis

SYMMETRIC CIPHER MODEL

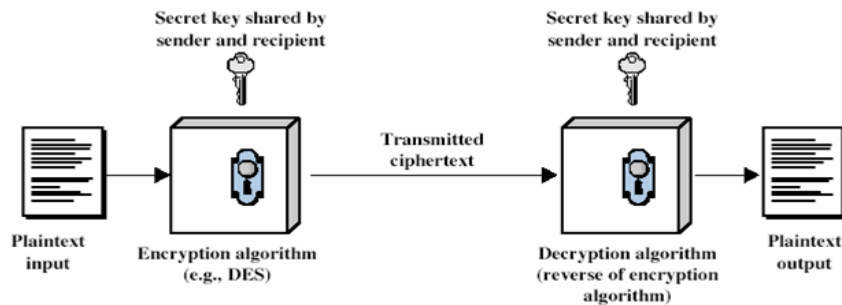
A symmetric encryption scheme has five ingredients:

- **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.
- **Ciphertext:** 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 ciphertexts. The ciphertext 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 ciphertext and the secret key and produces the original plaintext.

Two requirements for secure use of symmetric encryption:

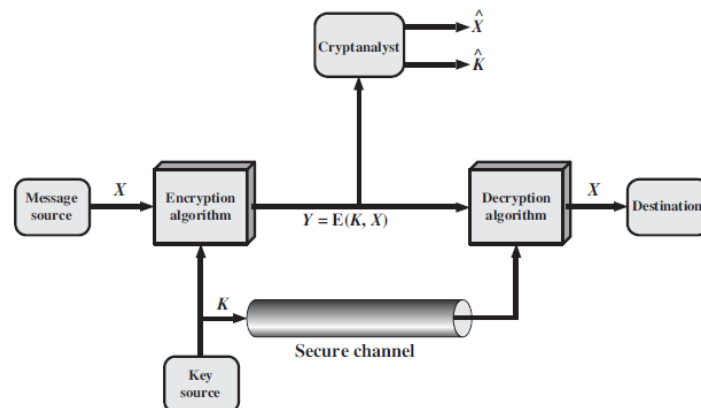
Information Security unit-I

- a strong encryption algorithm
 - Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure.
- If someone can discover the key and knows the algorithm, all communication using this key is readable.



We assume that it is impractical to decrypt a message on the basis of the ciphertext *plus* knowledge of the encryption/decryption algorithm. In other words, we do not need to keep the algorithm secret; we need to keep only the key secret. This feature of symmetric encryption is what makes it feasible for widespread use.

If the key is generated at the message source, then it must also be provided to the destination by means of some secure channel. Alternatively, a third party could generate the key and securely deliver it to both source and destination.



Cryptography:

Cryptographic systems are generally classified along 3 independent dimensions:

Type of operations used for transforming plain text to cipher text:

All the encryption algorithms are based on two general principles: **substitution**, in which each element in the plaintext is mapped into another element, and **transposition**, in which elements in the plaintext are rearranged.

The number of keys used:

If the sender and receiver uses same key then it is said to be **symmetric key (or) single key (or) conventional encryption**. If the sender and receiver use different keys then it is said to be **public key encryption**.

The way in which the plain text is processed:

A *block cipher* processes the input one block of elements at a time, producing an output block for each input block.

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

Cryptanalysis:

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

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

Cipher text only – A copy of cipher text alone is known to the cryptanalyst.

Known plaintext – The cryptanalyst has a copy of the cipher text and the corresponding plaintext.

Chosen plaintext – The cryptanalysts gains temporary access to the encryption machine. They cannot open it to find the key, however; they can encrypt a large number of suitably chosen plaintexts and try to use the resulting cipher texts to deduce the key.

Chosen cipher text – The cryptanalyst obtains temporary access to the decryption machine, uses it to decrypt several string of symbols, and tries to use the results to deduce the key.

Cryptanalysis

Cryptanalysis: Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext–ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used.

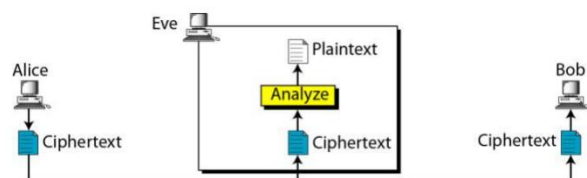
Brute-force attack: The attacker tries every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained. On average, half of all possible keys must be tried to achieve success.

Types of Attacks on Encrypted Messages

Ciphertext-only attack:

In a **ciphertext-only attack**, Eve has access to only some ciphertext. She tries to find the corresponding key and the plaintext. The ciphertext-only attack is the most probable one because Eve needs only the ciphertext for this attack.

Figure 3.4 Ciphertext-only attack



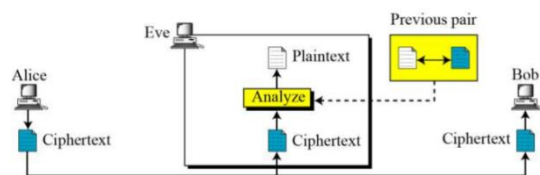
Known-Plaintext Attack:

In a **known-plaintext attack**, Eve has access to some plaintext/ciphertext pairs in addition to the intercepted ciphertext that she wants to break.

The plaintext/ciphertext pairs have been collected earlier. For example, Alice has sent a secret message to Bob, but she has later made the contents of the message public.

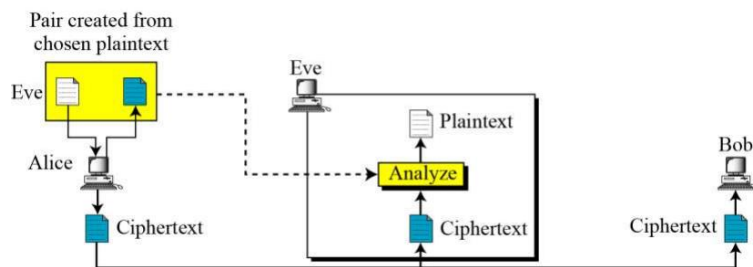
This attack is easier to implement because Eve has more information to use for analysis. However, it is less likely to happen because Alice may have changed her key or may have not disclosed the content of any previous messages.

Figure 3.5 Known-plaintext attack



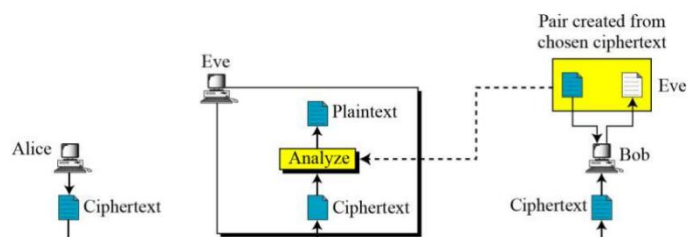
Chosen-Plaintext Attack:

The **chosen-plaintext attack** is similar to the known-plaintext attack, but the plaintext/ciphertext pairs have been chosen by the attacker herself. This can happen, for example, if Eve has access to Alice's computer. She can choose some plaintext and intercept the created ciphertext. This type of attack is much easier to implement, but it is much less likely to happen



Chosen-ciphertext Attack:

The **chosen-ciphertext attack** is similar to the chosen-plaintext attack, except that Eve chooses some ciphertext and decrypts it to form a ciphertext/plaintext pair. This can happen if Eve has access to Bob's computer



Type of Attack	Known to Cryptanalyst
Ciphertext Only	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext
Known Plaintext	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext • One or more plaintext–ciphertext pairs formed with the secret key
Chosen Plaintext	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key
Chosen Ciphertext	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key
Chosen Text	<ul style="list-style-type: none"> • Encryption algorithm • Ciphertext • Plaintext message chosen by cryptanalyst, together with its corresponding ciphertext generated with the secret key • Ciphertext chosen by cryptanalyst, together with its corresponding decrypted plaintext generated with the secret key

CLASSICAL ENCRYPTION TECHNIQUES

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

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

(i)**Caesar cipher (or) shift cipher** The earliest known use of a substitution cipher and the simplest was by Julius Caesar. The Caesar cipher involves replacing each letter of the alphabet with the letter standing 3 places further down the alphabet.

Plain : meet me after the toga party
cipher: 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:

When letters are involved, the following conventions are used in this book. Plaintext is always in lowercase; ciphertext is in uppercase; key values are in italicized lowercase.

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 ciphertext 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

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

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

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

Monoalphabetic Ciphers:

With only 25 possible keys, the Caesar cipher is far from secure. Hence, another approach referred to as monoalphabetic substitution cipher is introduced, in which instead of taking any permutation (possibility) of 26 characters then there are $26!$ or greater than possible keys.

Monoalphabetic cipher can use a key in different order. For example

Using the key “zebras” the ciphertext will be as follows.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
Z	E	B	R	A	S	C	D	F	G	H	I	J	K	L	M

Q	R	S	T	U	V	W	X	Y	Z
N	O	P	Q	T	U	V	W	X	Y

PlainText : hello

CipherText :DAILL

The cryptanalyst may try to break the ciphertext by looking at the ciphertext and identifying the frequency of letters in the text, comparing it with the relative frequency of the letters in English Text, and the looking at the frequency of two-letter word (digram), three letter words (trigram) and so on.

Information Security unit-I

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.

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.

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

The playfair algorithm is based on the use of 5x5 matrix of letters constructed using a keyword. Let the keyword be „monarchy“. The matrix is constructed by filling in the letters of the keyword (minus duplicates) from left to right and from top to bottom, and then filling in the remainder of the matrix with the remaining letters in alphabetical order.

The letter „i“ and „j“ count as one letter. Plaintext is encrypted two letters at a time according to the following rules:

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

Plaintext = meet me at the school house

Splitting two letters as a unit => me et me at th es ch ox ol ho us ex

Corresponding cipher text => CL KL CL RS PD IL HY AV MP HF XL IU

Strength of playfair cipher

- Playfair cipher is a great advance over simple mono alphabetic ciphers.
- Since there are 26 letters, $26 \times 26 = 676$ digrams are possible, so identification of individual digram is more difficult.

- Frequency analysis is much more difficult.

Hill Cipher:

The Hill Cipher was invented by Lester S. Hill in 1929, and like the other Digraphic Ciphers it acts on groups of letters. Unlike the others though it is extendable to work on different sized blocks of letters. So, technically it is a polygraphic substitution cipher, as it can work on digraphs, trigraphs (3 letter blocks) or theoretically any sized blocks.

The Hill Cipher uses an area of mathematics called Linear Algebra, and in particular requires the user to have an elementary understanding of matrices. It also makes use of Modulo Arithmetic (like the Affine Cipher). Because of this, the cipher has a significantly more mathematical nature than some of the others. However, it is this nature that allows it to act (relatively) easily on larger blocks of letters.

Encryption

To encrypt a message using the Hill Cipher we must first turn our keyword into a key matrix (a 2 x 2 matrix for working with digraphs, a 3 x 3 matrix for working with trigraphs, etc). We also turn the plaintext into digraphs (or trigraphs) and each of these into a column vector. We then perform matrix multiplication modulo the length of the alphabet (i.e. 26) on each vector. These vectors are then converted back into letters to produce the ciphertext.

2x2Example

We shall encrypt the plaintext message "short example" using the keyword *hill* and a 2 x 2 matrix. The first step is to turn the keyword into a matrix. If the keyword was longer than the 4 letters needed, we would only take the first 4 letters, and if it was shorter, we would fill it up with the alphabet in order (much like a [Mixed Alphabet](#)).

$$\begin{pmatrix} H & I \\ L & L \end{pmatrix}$$

The keyword written as a matrix.

With the keyword in a matrix, we need to convert this into a key matrix. We do this by converting each letter into a number by its position in the alphabet (starting at 0). So, A = 0, B = 1, C = 2, D = 3, etc.

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}$$

The key matrix (each letter of the keyword is converted to a number).

We now split the plaintext into digraphs, and write these as column vectors. That is, in the first column vector we write the first plaintext letter at the top, and the second letter at the bottom. Then we move to the next column vector, where the third plaintext letter goes at the top, and the fourth at the bottom. This continues for the whole plaintext.

$$\begin{pmatrix} s \\ h \end{pmatrix} \begin{pmatrix} o \\ r \end{pmatrix} \begin{pmatrix} t \\ e \end{pmatrix} \begin{pmatrix} x \\ a \end{pmatrix} \begin{pmatrix} m \\ p \end{pmatrix} \begin{pmatrix} l \\ e \end{pmatrix}$$

The plaintext "short example" split into column vectors.

Now we must convert the plaintext column vectors in the same way that we converted the keyword into the key matrix. Each letter is replaced by its appropriate number.

$$\begin{pmatrix} 18 \\ 7 \end{pmatrix} \begin{pmatrix} 14 \\ 17 \end{pmatrix} \begin{pmatrix} 19 \\ 4 \end{pmatrix} \begin{pmatrix} 23 \\ 0 \end{pmatrix} \begin{pmatrix} 12 \\ 15 \end{pmatrix} \begin{pmatrix} 11 \\ 4 \end{pmatrix}$$

The plaintext converted into numeric column vectors.

Now we must perform some matrix multiplication. We multiply the key matrix by each column vector in turn. We shall go through the first of these in detail, then the rest shall be presented in less detail. We write the key matrix first, followed by the column vector.

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 18 \\ 7 \end{pmatrix}$$

To perform matrix multiplication we "combine" the top row of the key matrix with the column vector to get the top element of the resulting column vector. We then "combine" the bottom row of the key matrix with the column vector to get the bottom element of the resulting column vector. The way we "combine" the four numbers to get a single number is that we multiply the first element of the key matrix row by the top element of the column vector, and multiply the second element of the key matrix row by the bottom element of the column vector. We then add together these two answers.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax + by \\ cx + dy \end{pmatrix}$$

That is, we follow the rules given by the algebraic method shown to the left.

The algebraic rules of matrix multiplication.

$$7 \times 18 + 8 \times 7 = 182$$

In our case we perform the two calculations on the right. We then right these two answers out in a column vector as shown below.

$$11 \times 18 + 11 \times 7 = 275$$

The calculations performed when doing a matrix multiplication.

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 18 \\ 7 \end{pmatrix} = \begin{pmatrix} 182 \\ 275 \end{pmatrix}$$

The shorthand for the matrix multiplication.

Next we have to take each of these numbers, in our resultant column vector, modulo 26 (remember that means divide by 26 and take the remainder).

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 18 \\ 7 \end{pmatrix} = \begin{pmatrix} 182 \\ 275 \end{pmatrix} = \begin{pmatrix} 0 \\ 15 \end{pmatrix} \text{ mod } 26$$

Reducing the resultant column vector modulo 26.

Finally we have to convert these numbers back to letters, so 0 becomes "A" and 15 becomes "P", and our first two letters of the ciphertext are "AP".

$$\begin{pmatrix} H & I \\ L & L \end{pmatrix} \begin{pmatrix} s \\ h \end{pmatrix} = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 18 \\ 7 \end{pmatrix} = \begin{pmatrix} 182 \\ 275 \end{pmatrix} = \begin{pmatrix} 0 \\ 15 \end{pmatrix} \text{ mod } 26 = \begin{pmatrix} A \\ P \end{pmatrix}$$

The whole calculation: converting to numbers; the matrix multiplication; reducing modulo 26; converting back to letters.

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 18 \\ 7 \end{pmatrix}$$

$$7 \times 18 + 8 \times 7 = 182$$

$$11 \times 18 + 11 \times 7 = 275$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 18 \\ 7 \end{pmatrix} = \begin{pmatrix} 182 \\ 275 \end{pmatrix}$$

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 18 \\ 7 \end{pmatrix} = \begin{pmatrix} 182 \\ 275 \end{pmatrix} = \begin{pmatrix} 0 \\ 15 \end{pmatrix} \text{ mod } 26$$

$$\begin{pmatrix} H & I \\ L & L \end{pmatrix} \begin{pmatrix} s \\ h \end{pmatrix} = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} \begin{pmatrix} 18 \\ 7 \end{pmatrix} = \begin{pmatrix} 182 \\ 275 \end{pmatrix} = \begin{pmatrix} 0 \\ 15 \end{pmatrix} \text{ mod } 26 = \begin{pmatrix} A \\ P \end{pmatrix}$$

This gives us a final ciphertext of "APADJ TFTWLFJ".

Decryption process:

To decrypt a ciphertext encoded using the Hill Cipher, we must find the inverse matrix. Once we have the inverse matrix, the process is the same as encrypting. That is we multiply the inverse key matrix by the column vectors that the ciphertext is split into, take the results modulo the length of the alphabet, and finally convert the numbers back to letters.

Since the majority of the process is the same as encryption, we are going to focus on finding the inverse key matrix (not an easy task), and will then skim quickly through the other steps (for more information see Encryption above).

In general, to find the inverse of the key matrix, we perform the calculation below, where K is the key matrix, d is the determinant of the key matrix and $adj(K)$ is the adjugate matrix of K .

$$K^{-1} = d^{-1} \times adj(K)$$

General method to calculate the inverse key matrix.

$$\begin{pmatrix} H & I \\ L & L \end{pmatrix}$$

2 x 2 Example

We shall decrypt the example above, so we are using the keyword *hill* and our ciphertext is "APADJ TFTWLFJ". We start by writing out the keyword as a matrix and converting this into a key matrix as for encryption. Now we must convert this to the inverse key matrix, for which there are several steps.

The keyword written as a matrix.

$$\begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}$$

The key matrix (each letter of the keyword is converted to a number).

Step 1 - Find the Multiplicative Inverse of the Determinant

The [determinant](#) is a number that relates directly to the entries of the matrix. It is found by multiplying the top left number by the bottom right number and subtracting from this the product of the top right number and the bottom left number. This is shown algebraically below. Note that the notation for determinant has straight lines instead of brackets around our matrix.

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

Algebraic method to calculate the determinant of a 2 x 2 matrix.

Once we have found this value, we need to take the number modulo 26. Below is the way to calculate the determinant for our example.

$$\begin{vmatrix} 7 & 8 \\ 11 & 11 \end{vmatrix} = 7 \times 11 - 8 \times 11 = -11 = 15 \text{ mod } 26$$

Calculating the determinant of our 2 x 2 key matrix.

We now have to find the [multiplicative inverse of the determinant working modulo](#) 26. That is, the number between 1 and 25 that gives an answer of 1 when we multiply it by the determinant. So, in this case, we are looking for the number that we need to multiply 15 by to get an answer of 1 modulo 26. There are algorithms to calculate this, but it is often easiest to use trial and error to find the inverse.

$$d d^{-1} = 1 \text{ mod } 26$$

If d is the determinant, then we are looking for the inverse of d.

$$15 \times x = 1 \text{ mod } 26$$

The multiplicative inverse is the number we multiply 15 by to get 1 modulo 26.

$$15 \times 7 = 105 = 1 \text{ mod } 26$$

This calculation gives us an answer of 1 modulo 26.

So the multiplicative inverse of the determinant modulo 26 is 7. We shall need this number later.

Step 2 - Find the Adjugate Matrix

The [adjugate matrix](#) is a matrix of the same size as the original. For a 2 x 2 matrix, this is fairly straightforward as it is just moving the elements to different positions and changing a couple of signs. That is, we swap the top left and bottom right numbers in the key matrix, and change the sign of the the top right and bottom left numbers. Algebraically this is given below.

$$\text{adj} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$$

The adjugate matrix of a 2 x 2 matrix.

Again, once we have these values we will need to take each of them modulo 26 (in particular, we need to add 26 to the negative values to get a number between 0 and 25. For our example we get the matrix below.

$$\text{adj} \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix} = \begin{pmatrix} 11 & -8 \\ -11 & 7 \end{pmatrix} = \begin{pmatrix} 11 & 18 \\ 15 & 7 \end{pmatrix}$$

The adjugate matrix of the key matrix.

Step 3 - Multiply the Multiplicative Inverse of the Determinant by the Adjugate Matrix

To get the inverse key matrix, we now multiply the inverse determinant (that was 7 in our case) from step 1 by each of the elements of the adjugate matrix from step 2. Then we take each of these answers modulo 26.

$$7 \times \begin{pmatrix} 11 & 18 \\ 15 & 7 \end{pmatrix} = \begin{pmatrix} 77 & 126 \\ 165 & 49 \end{pmatrix} = \begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix} \text{ mod } 26$$

Multiplying the multiplicative inverse of the determinant by the adjugate to get the inverse key matrix.

That is:

$$\text{if } K = \begin{pmatrix} 7 & 8 \\ 11 & 11 \end{pmatrix}, \text{ then } K^{-1} = \begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix}$$

Now we have the inverse key matrix, we have to convert the ciphertext into column vectors and multiply the inverse matrix by each column vector in turn, take the results modulo 26 and convert these back into letters to get the plaintext.

$$\begin{aligned} \begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix} \begin{pmatrix} A \\ P \end{pmatrix} &= \begin{pmatrix} 25 & 22 \\ 1 & 23 \end{pmatrix} \begin{pmatrix} 0 \\ 15 \end{pmatrix} \\ &= \begin{pmatrix} 25 \times 0 + 22 \times 15 \\ 1 \times 0 + 23 \times 15 \end{pmatrix} \\ &= \begin{pmatrix} 330 \\ 345 \end{pmatrix} \\ &= \begin{pmatrix} 18 \\ 7 \end{pmatrix} \text{ mod } 26 \\ &= \begin{pmatrix} s \\ h \end{pmatrix} \end{aligned}$$

The decryption of the first digraph.

We get back our plaintext of "short example".

Polyalphabetic Ciphers:

Another way to improve on the simple monoalphabetic technique is to use different monoalphabetic substitutions as one proceeds through the plaintext message. The general name for this approach is **polyalphabetic substitution cipher**.

All these techniques have the following features in common:

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

VIGENÈRE CIPHER The best known, and one of the simplest, polyalphabetic ciphers is the Vigenère cipher. In this scheme, the set of related monoalphabetic substitution rules consists of the 26 Caesar ciphers with shifts of 0 through 25. Each cipher is denoted by a key letter, which is the ciphertext letter that substitutes for the plaintext letter a.

Information Security unit-I

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

$$\begin{aligned} C &= C_0, C_1, C_2, \dots, C_{n-1} = E(K, P) = E[(k_0, k_1, k_2, \dots, k_{m-1}), (p_0, p_1, p_2, \dots, p_{n-1})] \\ &= (p_0 + k_0) \bmod 26, (p_1 + k_1) \bmod 26, \dots, (p_{m-1} + k_{m-1}) \bmod 26, \\ &\quad (p_m + k_0) \bmod 26, (p_{m+1} + k_1) \bmod 26, \dots, (p_{2m-1} + k_{m-1}) \bmod 26, \dots \end{aligned}$$

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

$$C_i = (p_i + k) \bmod 26$$

Similarly, decryption is a generalization of

$$p_i = (C_i - k) \bmod 26$$

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

Key : deceptivedeceptivedeceptive

Plaintext : wearediscoveredsaveyourself

Ciphertext : ZICVTWQNGRZGVTWAVZHCQYGLMGJ

Expressed numerically, we have the following result.

key	3	4	2	4	15	19	8	21	4	3	4	2	4	15
plaintext	22	4	0	17	4	3	8	18	2	14	21	4	17	4
ciphertext	25	8	2	21	19	22	16	13	6	17	25	6	21	19

key	19	8	21	4	3	4	2	4	15	19	8	21	4
plaintext	3	18	0	21	4	24	14	20	17	18	4	11	5
ciphertext	22	0	21	25	7	2	16	24	6	11	12	6	9

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

Autokey System:

Vigenère proposed what is referred to as an autokey system, in which a keyword is concatenated with the plaintext itself to provide a running key. For our example,

Key : deceptivewearediscoveredsav

Information Security unit-I

Plaintext : wearediscoveredsaveyourself

Ciphertext : ZICVTWQNGKZEIIGASXSTSLVVWLA

Even this scheme is vulnerable to cryptanalysis. Because the key and the plaintext share the same frequency distribution of letters, a statistical technique can be applied.

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

$$c_i = p_i \oplus k_i$$

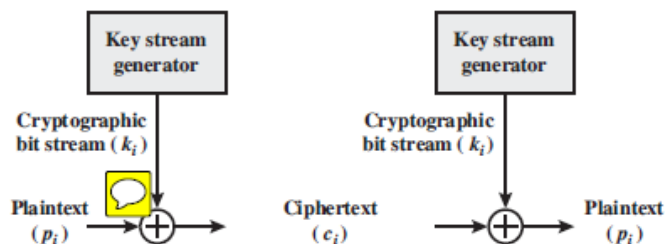
where

p_i = i th binary digit of plaintext

k_i = i th binary digit of key

c_i = i th binary digit of ciphertext

\oplus = exclusive-or (XOR) operation



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

$$p_i = c_i \oplus k_i$$

One-Time Pad

It is an improvement to the Vernam cipher that yields the ultimate in security. In this technique 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 ciphertext contains no information whatsoever about the plaintext, there is simply no way to break the code.

An example should illustrate our point. Suppose that we are using a 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 ciphertext

ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

We now show two different decryptions using two different keys:

Information Security unit-I

Ciphertext : ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

Key : pxlmvmsydozufyrvzwc tnlebnecvgdupahfzzlmnyih

Plaintext : mr mustard with the candlestick in the hall

Ciphertext : ANKYODKYUREPFJBYOJDSPLREYIUNOFDOIUERFPLUYTS

Key : mfugpmiydgaxgoufhklmhsqdgogtewbqfgyovuhwt

Plaintext : miss scarlet with the knife in the library

Two fundamental difficulties:

1. There is the practical problem of making large quantities of random keys.

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.

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

II .TRANSPOSITION TECHNIQUES

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

Rail fence is simplest of such cipher, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows.

Example

To encipher the message “meet me after the toga party” with a rail fence of depth 2, we write the following:

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

The encrypted message is

MEMATRHTGPRYETEFETEOAAT

Row Transposition Ciphers: A more complex scheme is to write the message in a rectangle, row by row, and read the message off, column by column, but permute the order of the columns. The order of the columns then becomes the key to the algorithm. For example

Key : 4 3 1 2 5 6 7
Plaintext : a t t a c k p
 o s t p o n e
 d u n t i l t
 w o a m x y z

Ciphertext: TTNAAPTMTSUOAODWCOIXKNLYPETZ

Information Security unit-I

Thus, in this example, the key is 4312567. To encrypt, start with the column 1, in this case column 3. Write down all the letters in that column. Proceed to column 4, which is labeled 2, then column 2, then column 1, then columns 5, 6, and 7.

A pure transposition cipher is easily recognized because it has the same letter frequencies as the original plaintext. The transposition cipher can be made significantly more secure by performing more than one stage of transposition. The result is more complex permutation that is not easily reconstructed.

```
Key    : 4 3 1 2 5 6 7
Input  : t t n a a p t
        m t s u o a o
        d w c o i x k
        n l y p e t z
Output : NSCYAUOPTTWLTMDNAOIEPAXTTOKZ
```

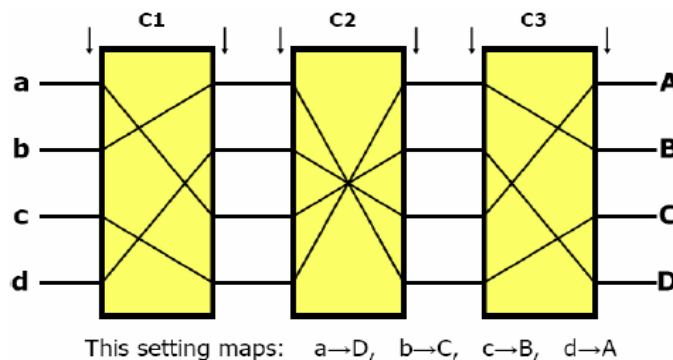
Rotor Machine:

Combine Substitution and Transposition Methods

- produce ciphers that are very difficult to break

Rotor Machines in World War II: German “Enigma” and Japanese “Purple”

- Breaking by the Allies was a significant factor in the outcome of the war (Turing)



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.

Drawbacks of steganography

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

BLOCK CIPHERS AND THE DATA ENCRYPTION STANDARD

BLOCK CIPHER PRINCIPLES:

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

Stream Ciphers and Block Ciphers:

A **stream cipher** is one that encrypts a digital data stream one bit or one byte at a time.

Examples of classical stream ciphers are the autokeyed Vigenère cipher and the Vernam cipher.

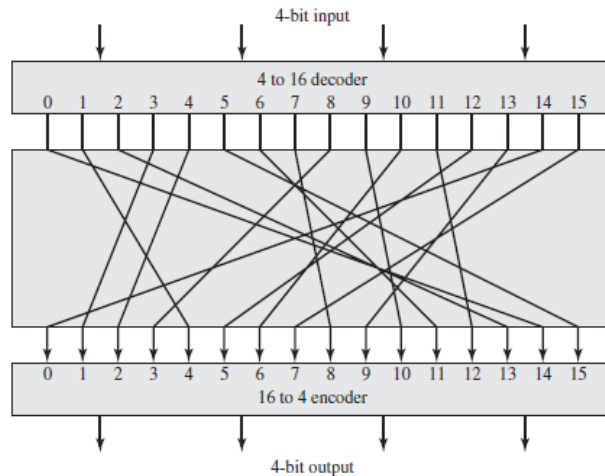
block cipher is one in which a block of plaintext is treated as a whole and used to produce a ciphertext block of equal length. Typically, a block size of 64 or 128 bits is used. A block cipher can be used to achieve the same effect as a stream cipher.

Motivation for the Feistel Cipher Structure:

A block cipher operates on a plaintext block of n bits to produce a ciphertext block of n bits. There are 2^n possible different plaintext blocks and, for the encryption to be reversible (i.e., for decryption to be possible), each must produce a unique ciphertext block. Such a transformation is called reversible, or nonsingular. The following examples illustrate nonsingular and singular transformations for $n = 2$.

Reversible Mapping		Irreversible Mapping	
Plaintext	Ciphertext	Plaintext	Ciphertext
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

In the latter case, a ciphertext of 01 could have been produced by one of two plaintext blocks. So if we limit ourselves to reversible mappings, the number of different transformations is $2^n!$.



The logic of a general substitution cipher for $n=4$. A 4-bit input produces one of 16 possible input states, which is mapped by the substitution cipher into a unique one of 16 possible output states, each of which is represented by 4 ciphertext bits. The encryption and decryption mappings can be defined by tabulation.

This is the most general form of block cipher and can be used to define any reversible mapping between plaintext and ciphertext. Feistel refers to this as the ideal block cipher, because it allows for the maximum number of possible encryption mappings from the plaintext block.

Plaintext	Ciphertext
0000	1110
0001	0100
0010	1101
0011	0001
0100	0010
0101	1111
0110	1011
0111	1000
1000	0011
1001	1010
1010	0110
1011	1100
1100	0101
1101	1001
1110	0000
1111	0111

Ciphertext	Plaintext
0000	1110
0001	0011
0010	0100
0011	1000
0100	0001
0101	1100
0110	1010
0111	1111
1000	0111
1001	1101
1010	1001
1011	0110
1100	1011
1101	0010
1110	0000
1111	0101

But there is a practical problem with the ideal block cipher. If a small block size, such as $n=4$, is used, then the system is equivalent to a classical substitution cipher. Such systems, as we have seen, are vulnerable to a statistical analysis of the plaintext. This weakness is not inherent in the use of a substitution cipher but rather results from the use of a small block size. If n is sufficiently large and an arbitrary reversible substitution between plaintext and ciphertext is allowed, then the statistical characteristics of the source plaintext are masked to such an extent that this type of cryptanalysis is infeasible.

The Feistel Cipher:

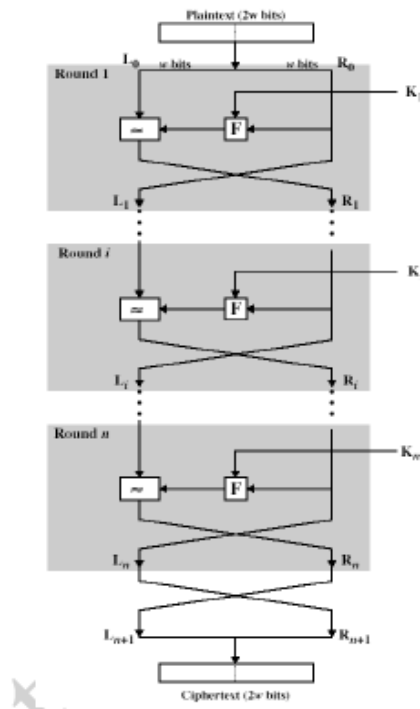
- **diffusion** – Statistical structure of the plaintext is dissipated into long-range statistics of the ciphertext
- **confusion** – make the relationship between the statistics of the ciphertext and the value of the encryption key as complex as possible

Feistel cipher structure:

The input to the encryption algorithm are a plaintext block of length $2w$ bits and a key K . The plaintext block is divided into two halves L_0 and R_0 . The two halves of the data pass through „ n “ rounds of processing and then combine to produce the ciphertext block. Each round „ i “ has inputs L_{i-1} and R_{i-1} , derived from the previous round, as well as the subkey K_i , derived from the overall key K . In general, the subkeys K_i are different from K and from each other.

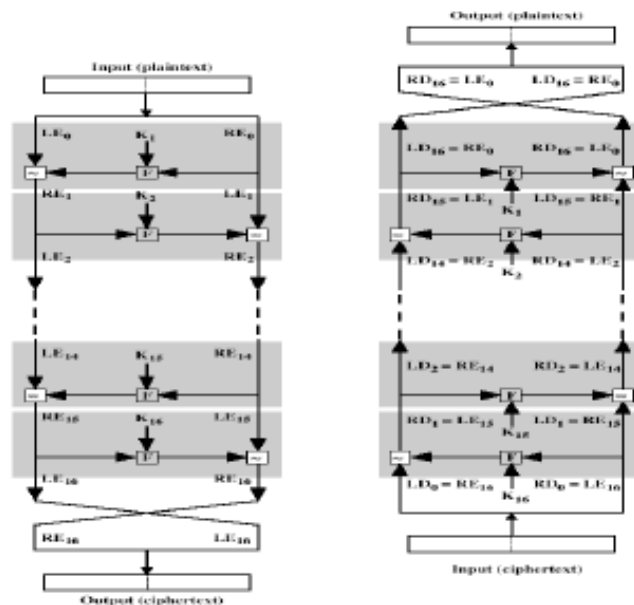
All rounds have the same structure. A substitution is performed on the left half of the data (as similar to S-DES). This is done by applying a round function F to the right half of the data and then taking the XOR of the output of that function and the left half of the data. The round function has the same general structure for each round but is parameterized by the round subkey k_i . Following this substitution, a permutation is performed that consists of the interchange of the two halves of the data. This structure is a particular form of the substitution-permutation network. The exact realization of a Feistel network depends on the choice of the following parameters and design features:

- **Block size:** Larger block sizes mean greater security (all other things being equal) but reduced encryption/decryption speed for a given algorithm. The greater security is achieved by greater diffusion. Traditionally, a block size of 64 bits has been considered a reasonable tradeoff and was nearly universal in block cipher design. However, the new AES uses a 128-bit block size.
- **Key size:** Larger key size means greater security but may decrease encryption/decryption speed. The greater security is achieved by greater resistance to brute-force attacks and greater confusion. Key sizes of 64 bits or less are now widely considered to be inadequate, and 128 bits has become a common size.
- **Number of rounds:** The essence of the Feistel cipher is that a single round offers inadequate security but that multiple rounds offer increasing security. A typical size is 16 rounds.
- **Subkey generation algorithm:** Greater complexity in this algorithm should lead to greater difficulty of cryptanalysis.
- **Round function F :** Again, greater complexity generally means greater resistance to cryptanalysis.



There are two other considerations in the design of a Feistel cipher:

- **Fast software encryption/decryption:** In many cases, encryption is embedded in applications or utility functions in such a way as to preclude a hardware implementation. Accordingly, the speed of execution of the algorithm becomes a concern.
- **Ease of analysis:** Although we would like to make our algorithm as difficult as possible to cryptanalyze, there is great benefit in making the algorithm easy to analyze.



The process of decryption is essentially the same as the encryption process. The rule is as follows: use the cipher text as input to the algorithm, but use the subkey k_i in reverse order. i.e., k_n in the first round, k_{n-1} in second round and so on. For clarity, we use the notation LE_i and RE_i for data traveling through the decryption algorithm. The diagram below indicates that, at each round, the intermediate value of the decryption process is same (equal) to the corresponding value of the encryption process with two halves of the value swapped. i.e.,

$$RE_i \parallel LE_i \text{ (or) equivalently } RD_{16-i} \parallel LD_{16-i}$$

After the last iteration of the encryption process, the two halves of the output are swapped, so that the cipher text is $RE_{16} \parallel LE_{16}$. The output of that round is the cipher text. Now take the cipher text and use it as input to the same algorithm. The input to the first round is $RE_{16} \parallel LE_{16}$, which is equal to the 32-bit swap of the output of the sixteenth round of the encryption process. Now we will see how the output of the first round of the decryption process is equal to a 32-bit swap of the input to the sixteenth round of the encryption process. First consider the encryption process,

$$LE_{16} = RE_{15}$$

$$RE_{16} = LE_{15} \oplus F(RE_{15}, K_{16})$$

On the decryption side,

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$

$$\begin{aligned} RD_1 &= LD_0 \oplus F(RD_0, K_{16}) \\ &= RE_{16} \oplus F(RE_{15}, K_{16}) \\ &= [LE_{15} \oplus F(RE_{15}, K_{16})] \oplus F(RE_{15}, K_{16}) \\ &= LE_{15} \end{aligned}$$

Therefore,

$$LD_1 = RE_{15}$$

$$RD_1 = LE_{15}$$

In general, for the i^{th} iteration of the encryption algorithm,

$$LE_i = RE_{i-1}$$

$$RE_i = LE_{i-1} \oplus F(RE_{i-1}, K_i)$$

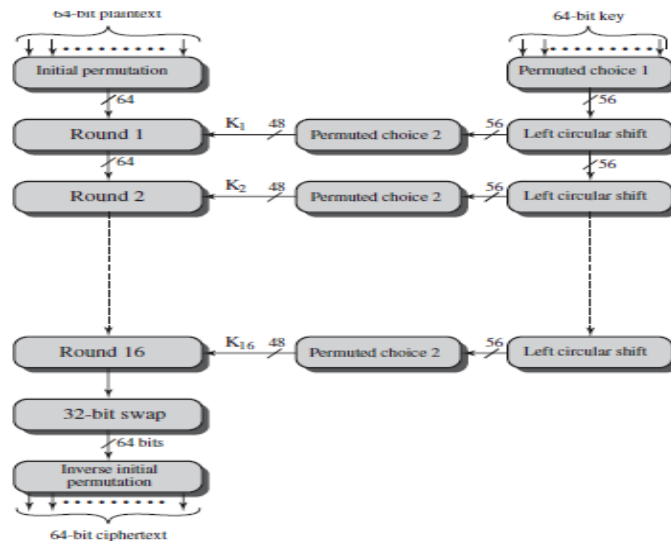
Finally, the output of the last round of the decryption process is $RE_0 \parallel LE_0$. A 32-bit swap recovers the original plaintext.

DATA ENCRYPTION STANDARD (DES)

The most widely used encryption scheme is based on the Data Encryption Standard (DES) adopted in 1977 by the National Bureau of Standards, now the National Institute of Standards and Technology (NIST), as Federal Information Processing Standard 46 (FIPS PUB 46). The algorithm itself is referred to as the Data Encryption Algorithm (DEA). For DES, data are encrypted in 64-bit blocks using a 56-bit key. The algorithm transforms 64-bit input in a series of steps into a 64-bit output. The same steps, with the same key, are used to reverse the encryption. The DES enjoys widespread use. It has also been the subject of much controversy concerning how secure the DES is.

DES Encryption:

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



Looking at the left-hand side of the figure, we can see that the processing of the plaintext proceeds in three phases. First, the 64-bit plaintext passes through an initial permutation (IP) that rearranges the bits to produce the *permuted input*. This is followed by a phase consisting of sixteen rounds of the same function, which involves both permutation and substitution functions. The output of the last (sixteenth) round consists of 64 bits that are a function of the input plaintext and the key. The left and right halves of the output are swapped to produce the **preoutput**. Finally, the preoutput is passed through a permutation [IP-1] that is the inverse of the initial permutation function, to produce the 64-bit ciphertext.

The right-hand portion of above Figure shows the way in which the 56-bit key is used. Initially, the key is passed through a permutation function. Then, for each of the sixteen rounds, a *subkey* (K_i) is produced by the combination of a left circular shift and a permutation. The permutation function is the same for each round, but a different subkey is produced because of the repeated shifts of the key bits.

INITIAL PERMUTATION The initial permutation and its inverse are defined by tables. The tables are to be interpreted as follows. The input to a table consists of 64 bits numbered from 1 to 64. The 64 entries in the permutation table contain a permutation of the numbers from 1 to 64. Each entry in the permutation table indicates the position of a numbered input bit in the output, which also consists of 64 bits.

To see that these two permutation functions are indeed the inverse of each other, consider the following 64-bit input M:

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

where M_i is a binary digit. Then the permutation $X = (IP(M))$ is as follows:

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

If we then take the inverse permutation $Y = IP^{-1}(X) = IP^{-1}(IP(M))$, it can be seen that the original ordering of the bits is restored.

(a) Initial Permutation (IP)

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

(b) Inverse Initial Permutation (IP^{-1})

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

(c) Expansion Permutation (E)

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

- Initially the key is passed through a permutation function (PC_1)
- For each of the 16 iterations, a subkey (K_i) is produced by a combination of a left circular shift and a permutation (

Information Security unit-I

PC₂) which is the same for each iteration. However, the resulting subkey is different for each iteration because of repeated shifts

(a) Input Key							
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

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

(c) Permuted Choice Two (PC-2)							
14	17	31	24	1	5	3	28
15	6	23	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

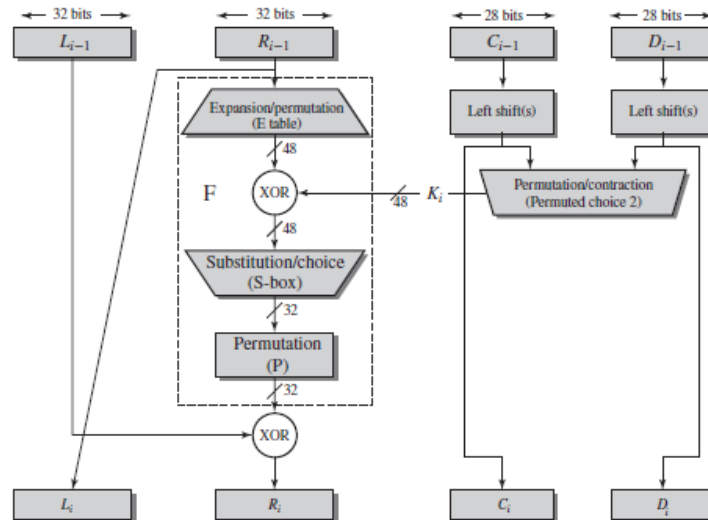
(d) Schedule of Left Shifts															
Round Number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Bits Rotated	1	1	2	2	2	2	2	1	2	2	2	2	2	2	1

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

DETAILS OF SINGLE ROUND Below Figure shows the internal structure of a single round. Again, begin by focusing on the left-hand side of the diagram. The left and right halves of each 64-bit intermediate value are treated as separate 32-bit quantities, labeled L (left) and R (right). As in any classic Feistel cipher, the overall processing at each round can be summarized in the following formulas

$$L_i = R_{i-1}$$

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



The E-box expansion permutation - here the 32-bit input data from R_{i-1} is expanded and permuted to give the 48 bits necessary for combination with the 48 bit key (defined in table 2.1). The E-box expansion permutation delivers a larger output by splitting its input into 8, 4-bit blocks and copying every first and fourth bit in each block into the output in a defined manner. The security offered by this operation comes from one bit affecting two substitutions in the S-boxes. This causes the dependency of the output bits on the input bits to spread faster, and is known as the avalanche affect.

2. The bit by bit addition modulo 2 (or exclusive OR) of the E-box output and 48 bit subkey K_i .
3. The S-box substitution - this is a highly important substitution which accepts a 48-bit input and outputs a 32-bit number (defined in table 2.3). The S-boxes are the only non-linear operation in DES and are therefore the most important part of its security. They were very carefully designed although the conditions they were designed under has been under intense scrutiny since DES was released. The reason was because IBM had already designed a set of S-boxes which were completely changed by the NSA with no explanation why.

The input to the S-boxes is 48 bits long arranged into 8, 6 bit blocks (b_1, b_2, \dots, b_6). There are 8 S-boxes (S_1, S_2, \dots, S_8) each of which accepts one of the 6 bit blocks. The output of each S-box is a four bit number. Each of the S-boxes can be thought of as a 4×16 matrix. Each cell of the matrix is identified by a coordinate pair (i, j) , where $0 \leq i \leq 3$ and $0 \leq j \leq 15$. The value of i is taken as the decimal representation of the first and last bits of the input to each S-box, i.e. $\text{Dec}(b_1 b_6) = i$ and the value of j is take from the decimal representation of the inner four bits that remain, S-box matrices contains a 4-bit number which is output once that particular cell is selected by the input.

4. The P-box permutation - This simply permutes the output of the S-box without changing the size of the data (defined in table 2.1). It is simply a permutation and nothing else. It has a one to one mapping of its input to its output giving a 32 bit output from a 32 bit input.

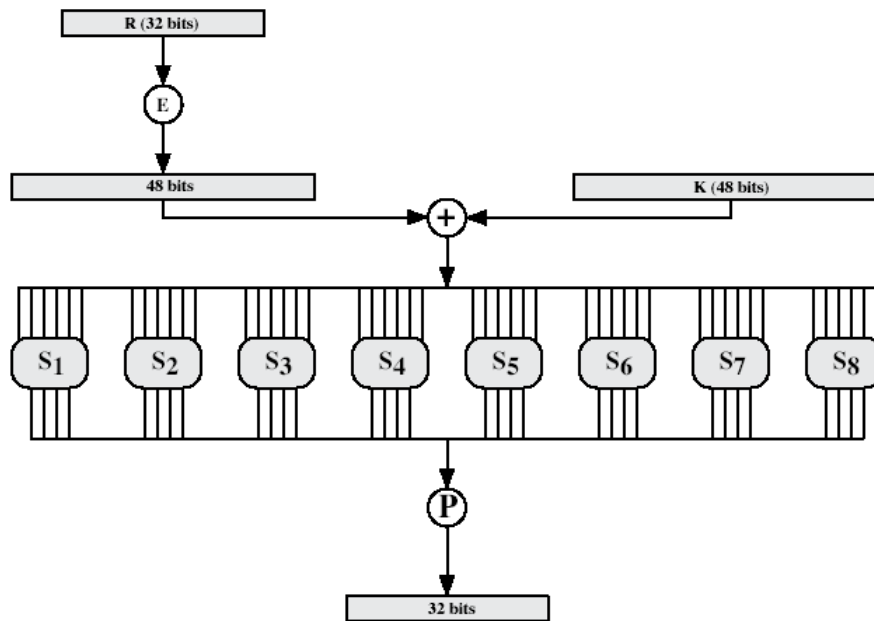


Figure 3.9 Calculation of $F(R, K)$

Information Security unit-I

Table 3.3 Definition of DES S-Boxes

S_1	14 4 13 1 2 15 11 8 3 10 6 12 5 9 0 7 0 15 7 4 14 2 13 1 10 6 12 11 9 5 3 8 4 1 14 8 13 6 2 11 15 12 9 7 5 10 5 0 15 12 8 2 4 9 1 7 6 11 3 14 10 0 6 13
S_2	15 1 8 14 5 11 3 4 9 7 2 13 12 0 5 10 3 13 4 7 15 2 8 14 12 0 1 10 6 9 11 5 0 14 7 11 10 4 13 1 5 8 12 6 9 3 2 15 13 8 10 1 3 15 4 2 11 6 7 12 0 5 14 9
S_3	10 0 9 14 6 3 15 5 1 13 12 7 11 4 2 8 13 7 0 9 3 4 6 10 2 8 5 14 12 11 15 1 13 8 4 9 8 15 3 0 11 1 2 12 5 10 14 7 1 10 13 0 6 9 8 7 4 15 14 5 11 5 2 12
S_4	7 13 14 3 0 6 9 10 1 2 8 5 11 12 4 15 13 8 11 5 6 15 0 3 4 7 2 12 1 10 14 9 10 8 9 0 12 11 7 13 15 1 3 14 5 2 8 4 3 15 0 6 10 1 13 8 9 4 5 11 12 7 2 14
S_5	2 12 4 1 7 10 11 6 8 5 3 15 13 0 14 9 14 11 2 13 4 7 13 1 5 0 15 10 3 9 8 6 4 2 1 11 10 13 7 8 15 9 12 5 6 3 0 14 11 8 12 7 1 14 2 13 6 15 0 9 10 4 5 3
S_6	12 1 10 15 9 2 6 8 0 13 3 4 14 7 5 11 10 15 4 2 7 12 9 5 6 1 13 14 0 11 3 8 9 14 15 5 2 8 12 3 7 0 4 10 1 13 11 6 4 3 2 12 9 5 15 10 11 14 1 7 6 0 8 13
S_7	4 11 2 14 15 0 8 13 3 12 9 7 5 10 6 1 13 0 11 7 4 9 1 10 14 3 5 12 2 15 8 6 1 4 11 13 12 3 7 14 10 15 6 8 0 5 9 2 6 11 13 8 1 4 10 7 9 5 0 15 14 2 3 12
S_8	13 2 8 4 6 15 11 1 10 9 3 14 5 0 12 7 1 15 13 8 10 3 7 4 12 5 6 11 0 14 9 2 7 11 4 1 9 12 14 2 0 6 10 13 15 3 5 8 2 1 14 7 4 10 8 13 15 12 9 0 3 5 6 11

Each row of an S-box defines a general reversible substitution: middle 4 bits of each group of 6-bit input are substituted by S-box output, 1st and last 6th bits define what particular substitution out of to use.

DES decryption:

As with any feistel cipher, decryption uses the same algorithm as encryption, except that the application of the subkeys is reverse.

Avalanche effect

A desirable property of any encryption algorithm is that a small change in either plaintext or key should produce significant changes in the ciphertext. DES exhibits a strong avalanche effect.

(a) Change in Plaintext		(b) Change in Key	
Round	Number of bits that differ	Round	Number of bits that differ
0	1	0	0
1	6	1	2
2	21	2	14
3	35	3	28
4	39	4	32
5	34	5	30
6	32	6	32
7	31	7	35
8	29	8	34
9	42	9	40
10	44	10	38
11	32	11	31
12	30	12	33
13	30	13	28
14	26	14	26
15	29	15	34
16	34	16	35

Avalanche effect - a small change in the plaintext produces a significant change in the ciphertext.

Strength of DES

The Use of 56-Bit Keys

- With a key length of 56 bits, there are 2^{56} possible keys, which is approximately 7.2×10^{16} . Brute force search looks hard.
- DES finally and definitively proved insecure in July 1998, when the Electronic Frontier Foundation (EFF) announced that it had broken a DES encryption using a special-purpose “DES cracker” machine. The attack took less than three days.
- Alternatives to DES are required-which are AES and triple DES.

The Nature of the DES Algorithm

- Another concern is the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm.
- The focus of concern has been on the eight substitution tables, or S-boxes, that are used in each iteration.
- No one has so far succeeded in discovering the supposed fatal weaknesses in the S-boxes.

Timing Attacks

- A timing attack is one in which information about the key or the plaintext is obtained by observing how long it takes a given implementation to perform decryptions on various cipher texts.
- A timing attack exploits the fact that an encryption or decryption algorithm often takes slightly different amounts of time on different inputs.
- DES is resistant to a successful timing attack.

Cryptanalytic attacks on DES

- **Differential cryptanalysis**

- Initially- less than 2^{55} encryptions
- cryptanalyze DES with an effort on the order of 2^{47} encryptions, requiring 2^{47} chosen plaintexts.
- Differential cryptanalysis is to observe the behavior of pairs of text blocks evolving along each round of the cipher.
- Many pairs of inputs to f with the same difference yield the same output difference if the same subkey is used.
- Linear analysis-This method can find a DES key given known 2^{43} plaintexts, as compared to 2^{47} chosen plaintexts for differential cryptanalysis.
- The need to strengthen DES against attacks using differential cryptanalysis played a large part in the design of the S-boxes and the permutation P .
- Differential cryptanalysis of an eight-round LUCIFER algorithm requires only 256 chosen plaintexts, whereas an attack on an eight-round version of DES requires 2^{14} chosen plaintexts.
- **Linear cryptanalysis**

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

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

•