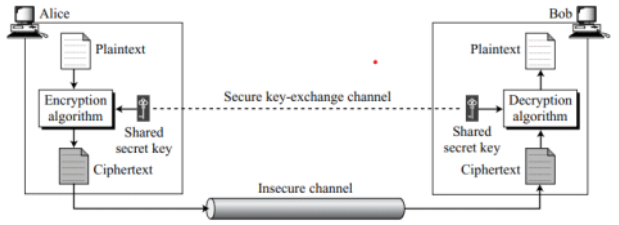
Introduction :-

Figure 3.1 *General idea of symmetric-key cipher*



The original message from Alice to Bob is called **plaintext;** the message that is sent through the channel is called the **ciphertext.** To create the ciphertext from the plaintext, Alice uses an **encryption algorithm** and a **shared secret key.** To create the plaintext from ciphertext, Bob uses a **decryption algorithm** and the same secret key. We refer to encryption and decryption algorithms as **ciphers.** A **key** is a set of values (numbers) that the cipher, as an algorithm, operates on.

01. Alice wants to send some msg to Bob ⇒ plaintext

Imp :- ① A single key is used for encryption and decryption
② Encryption and Decryption algorithms are inverse of each other.

Encryption ⇒ $E_K(x)$ Decryption = $D_K(x)$

$\therefore D_K(E_K(x)) = E_K(D_K(x)) = x$

M people in a group communicate with each other

No. of keys required = $\frac{m(m-1)}{2}$

Kerckhoff's Principle

Although it may appear that a cipher would be more secure if we hide both the encryption/decryption algorithm and the secret key, this is not recommended. Based on **Kerckhoff's principle,** one should always assume that the adversary, Eve, knows the encryption/decryption algorithm. The resistance of the cipher to attack must be based only on the secrecy of the key. In other words, guessing the key should be so difficult that there is no need to hide the encryption/decryption algorithm. This principle manifests itself more clearly when we study modern ciphers. There are only a few algorithms for modern ciphers today. The **key domain** for each algorithm, however, is so large that it makes it difficult for the adversary to find the key.

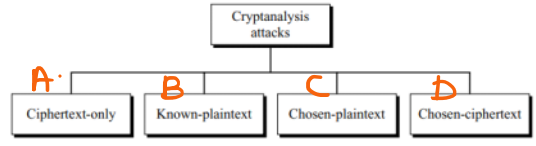
Cryptanalysis →

① cryptography → science and art of creating secret codes

② cryptanalysis → science and art of breaking those codes

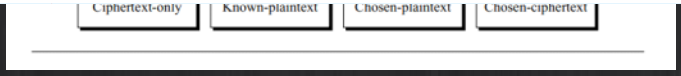
This is needed not to break others code but to know how vulnerable our system is.

Figure 3.3 *Cryptanalysis attacks*



A. Cipher text-only attack :-

01. Attacker have some access to ciphertext
02. The attacker tries to find corresponding key and plaintext.



02. The attacker tries to find corresponding key and plaintext.
03. Assumption that the attacker knows the algorithm and can intercept the cipher text.

Ciphertext-only
- Brute force or Exhaustive search
- Statistical Attack
- Pattern Attack.

a. Bruteforce or Exhaustive search :-

01. The attacker tries to use all the possible keys.
We assume that the attacker knows the algorithm and the domain of keys (all possible keys)

Ciphertext-Only Attack: Example
- Wired Equivalent Privacy (WEP), the first security protocol for Wi-Fi was proved vulnerable ciphertext only attack.

b. Statistical Attack :-

01. The cryptanalyst can benefit from the inherent characteristics of the plaintext language to launch a statistical attack.

Statistical Attack

The cryptanalyst can benefit from some inherent characteristics of the plaintext language to launch a **statistical attack.** For example, we know that the letter E is the most-frequently used letter in English text. The cryptanalyst finds the mostly-used character in the ciphertext and assumes that the corresponding plaintext character is E. After finding a few pairs, the analyst can find the key and use it to decrypt the message. To prevent this type of attack, the cipher should hide the characteristics of the language.

Pattern Attack

Some ciphers may hide the characteristics of the language, but may create some patterns in the ciphertext. A cryptanalyst may use a **pattern attack** to break the cipher. Therefore, it is important to use ciphers that make the ciphertext look as random as possible.

c. Pattern Attack :-

01. Hides the characteristics but creates some patterns for the ciphertext. This allows the attacker to use the pattern attack.

B. Known Plain Text Attack

=> 01. The attacker has some access to the plaintext/ciphertext apart from the intercepted ciphertext.
02. The main goal of the attacker is to use this information to discover the keys used for encryption decryption.

C. Chosen Plain Text Attack

=> 01. The attacker chooses the plaintext ciphertext herself.

02. **for example**, if Eve has access to Alice's computer. She can choose some plaintext and intercept the created ciphertext. Of course, she does not have the key because the key is normally embedded in the software used by the sender. This type of attack is much easier to implement, but it is much less likely to happen.

D. Choosen Cipher Text Attack

=> 01. Similar to C but the attacker chooses the cipher text and decrypts it to form ciphertext/plaintext

This can happen if Eve has access to Bob's computer

* Substitution Cipher:- 01. replaces one symbol with another

Monoalphabetic Ciphers

We first discuss a group of substitution ciphers called the **monoalphabetic ciphers.** In monoalphabetic substitution, a character (or a symbol) in the plaintext is always changed to the same character (or symbol) in the ciphertext regardless of its position in the text. For example, if the algorithm says that letter A in the plaintext is changed to letter D, every letter A is changed to letter D. In other words, the relationship between letters in the plaintext and the ciphertext is one-to-one.

⇒ex plaintext :- HELLO
ciphertext :- KHOOR

Case I :- Plaintext = 'HELLO' Ciphertext :- 'KHOOR' - monoalphabetic

Case II :- Plaintext = 'HELLO' Ciphertext = 'ABNZF' - Not monoalphabetic

Additive Cipher

Additive Cipher:-
01. simplest monoalphabetic cipher - additive cipher.
02. Sometime known as Shift Cipher, sometimes Caesar cipher.
03. plaintext = lowercase characters 'a' to 'z'
ciphertext = uppercase characters 'A' to 'Z'
Assigning values to each character (upper case and lower)
↳ (0-25) - b/w z (25 = z) last character

Figure 3.8 *Representation of plaintext and ciphertext characters in* Z_{26}

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	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
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

$$C = (P + K) \bmod 26$$
$$P = (C - K) \bmod 26$$

P = Plaintext
C = Ciphertext

The encryption algorithm adds the key to plaintext character — the decryption algorithm subtracts the key from the cipher text character.

P = plaintext C = ciphertext $C = (P+K) \bmod 26 \rightarrow ①$
P_1 = plaintext created by Bob

$P_1 = (C-K) \bmod 26$
from ①
$P_1 = [(P+K) - K] \bmod 26$
$P \bmod 26$
$P_1 = P$.

Example:-

example Use additive cipher with key=15 to encrypt message "hello"
solution

Plaintext = h = 11 Encryption = (11+15) mod 26 Ciphertext = 00 = A
o = 14 Encryption = (14+15) mod 26
Ciphertext = 03 → D

result :- Ciphertext = WTAAD.

"hello"

solution

Plaintext :- h=07

Encryption = (07+15) mod 26

Ciphertext = 22 = W

Plaintext : e = 04

Encryption = (04+15) mod 26

Ciphertext = 19 = T

result :- Ciphertext = WTAAD.

Example 3.4

Use the additive cipher with key = 15 to decrypt the message "WTAAD".

formula :- P = (C-K) mod 26

Solution :- Ciphertext W = 22

T = 19

A = 00

Plaintext = (22-15) mod 26 = 7 = h

= (19-15) mod 26 = 4 = e

(0 -15) mod 26 = -15

(26-15) mod 26

= 11 = l

D = 03 Plaintext = (3-15) mod 26

= (-12) mod 26

= (26-12) mod 26

= 14 = o

WTAAD → hello

Shift Cipher

Historically, additive ciphers are called shift ciphers. The reason is that the encryption algorithm can be interpreted as "shift *key* characters down" and the encryption algorithm can be interpreted as "shift *key* character up". For example, if the key = 15, the encryption algorithm shifts 15 characters down (toward the end of the alphabet). The decryption algorithm shifts 15 characters up (toward the beginning of the alphabet). Of course, when we reach the end or the beginning of the alphabet, we wrap around (manifestation of modulo 26).

Plaintext →	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext →	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
Value →	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

→ shifting keys by 15

15 keys

⇒ W = h

Cryptanalysis :- 01. Additive Ciphers are only vulnerable to cipher-text only attacks using exhaustive key searches (brute force)

02. Key domain ⇒ very less (0-25) → 26 keys
but 0 is useless
↳ So only 25 keys.

Example 3.5

Eve has intercepted the ciphertext "UVACLYFZLJBYL". Show how she can use a brute-force attack to break the cipher.

formula :- $p = (c-k) \bmod 26$

Check for K

K=1 ⇒ for U ⇒ $(20-1) \bmod 26 = t$
for V ⇒ $(21-1) \bmod 26 = u$
for A ⇒ $(0-1) \bmod 26 = z$
for C ⇒ $(2-1) \bmod 26 = b$
⋮

Plaintext: tuzbkxeykiaxk

At the end we get

Additive ciphers are also subject to statistical attacks. This is especially true if the adversary has a long ciphertext. The adversary can use the frequency of occurrence of characters for a particular language. Table 3.1 shows the frequency for an English text of 100 characters.

Table 3.1 *Frequency of occurrence of letters in an English text*

Letter	*Frequency*	*Letter*	*Frequency*	*Letter*	*Frequency*	*Letter*	*Frequency*
E	12.7	H	6.1	W	2.3	K	0.08
T	9.1	R	6.0	F	2.2	J	0.02
A	8.2	D	4.3	G	2.0	Q	0.01
O	7.5	L	4.0	Y	2.0	X	0.01
I	7.0	C	2.8	P	1.9	Z	0.01
N	6.7	U	2.8	B	1.5		
S	6.3	M	2.4	V	1.0		

Frequency Analysis Attack Ex:-

Example 3.6

Eve has intercepted the following ciphertext. Using a statistical attack, find the plaintext.

XLILSYWIMWRSAJSVWEPIJSVJSYVQMPPMSRHSPPEVWMXMWASVX-LQSVILY-VVCFIJSVIXLIWIPPIVVIGIMZIWQSVISJJIVW

⇒ 01. In this particular ques occurence of I is the max I=14 V=13 times

02. It means that probably I is mapped to the letter 'e' from the plaintext.

'e' from the plaintext.

03. Consider the idea of shift ciphers.

वो है तो 'e' ही lekin map किया है 'I' से

e = 4 I = 8 key shifting ⇒ 8 - 4 = 4

Hence key = 4

Multiplicative Ciphers :-

01. In multiplicative cipher the encryption algorithm specifies multiplication of plain text by key.

decryption algorithm specifies division of ciphertext by key

Inshort ⇒ decryption में ciphertext का multiplication होता है with the inverse of the key.

Encryption :- $C = (P \cdot K) \bmod 26$

Decryption :- $C = (P \cdot K^{-1}) \bmod 26$

> In a multiplicative cipher, the plaintext and ciphertext are integers in Z_{26}; the key is an integer in Z_{26}^*.

Example 3.8

We use a multiplicative cipher to encrypt the message "hello" with a key of 7. The ciphertext is "XCZZU".

⇒ h = 07 key = 7

$C = (P \cdot K) \bmod 26$

$\Rightarrow C = (7 \cdot 7) \bmod 26$

$= 49 \bmod 26$

$= 23 \longrightarrow$ 'X'

'e' = 4 $C = (P \cdot K) \bmod 26$

$= C = (4 \cdot 7) \bmod 26$

$\Rightarrow C = (28) \bmod 26$

$\Rightarrow C = 2$ = 'c'

Similarly :-

Plaintext: h → 07	Encryption: (07 × 07) mod 26	ciphertext: 23 → X
Plaintext: e → 04	Encryption: (04 × 07) mod 26	ciphertext: 02 → C
Plaintext: l → 11	Encryption: (11 × 07) mod 26	ciphertext: 25 → Z
Plaintext: l → 11	Encryption: (11 × 07) mod 26	ciphertext: 25 → Z
Plaintext: o → 14	Encryption: (14 × 07) mod 26	ciphertext: 20 → U

Important :- what is the key domain for mulplicative Cipher?

Important :- what is the key domain for multiplicative Cipher?

⇒ The key needs to be in Z_{26}^*

i.e. the numbers lying b/w (0-25) → should have inverse.

⇒ only 12 numbers are present in the key domain

{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25}

Affine Ciphers:-

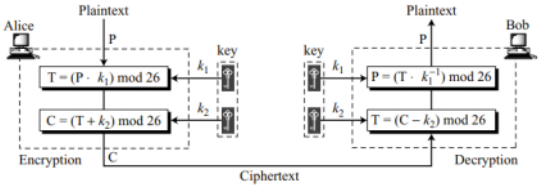
01. Combination of additive and multiplicative Ciphers.

02. 2 keys are used
   (a) first key is used with multiplicative cipher
   (b) second key is used with additive cipher.

Affine Cipher

We can combine the additive and multiplicative ciphers to get what is called the **affine cipher**—a combination of both ciphers with a pair of keys. The first key is used with the multiplicative cipher; the second key is used with the additive cipher. Figure 3.11 shows that the affine cipher is actually two ciphers, applied one after another. We could have shown only one complex operation for the encryption or decryption such as $C = (P \times k_1 + k_2) \bmod 26$ and $P = ((C - k_2) \times k_1^{-1}) \bmod 26$. However, we have used a temporary result (T) and have indicated two separate operations to show that whenever we use a combination of ciphers we should be sure that each one has an inverse at the other side of the line and that they are used in reverse order in the encryption and decryption. If addition is the last operation in encryption, then subtraction should be the first in decryption.

Figure 3.11 *Affine cipher*



Combining Both :-

$$C = (P \times K_1 + K_2) \bmod 26$$

$$P = [(C - K_2) \cdot K_1^{-1}] \bmod 26$$

Question : key domain of Affine Cipher?

Soln first key = multiplicative Cipher = Z_{26}^* ⇒ 12 (proved above)

Second key = additive Cipher = Z_{26} = 26 ⇒ (0-25)

Total = 12 × 26 = 312

Example 3.10

Use an affine cipher to encrypt the message "hello" with the key pair (7, 2).

⇒ **Soln** $C = (P \times K_1 + K_2) \bmod 26$

⇒ hello ⇒ 'h' = 7 $C = (7 \times 7 + 2) \bmod 26 = 25 = 'Z'$

'e' = 4 $C = (4 \times 7 + 2) \bmod 26 = 4^? = 'E'$

'l' = 11 $C = (11 \times 7 + 2) \bmod 26 = 1 = 'B'$

'o' = 14 $C = (14 \times 7 + 2) \bmod 26 = 22 = 'W'$

C = ZEBBW

Example 3.11

Use the affine cipher to decrypt the message "ZEBBW" with the key pair (7, 2) in modulus 26.

formula ⇒ $P = ((C - k_2) \times k_1^{-1}) \bmod 26$

Soln $'Z' = 25 \Rightarrow P = ((25-2) \times 7^{-1}) \bmod 26$

Inverse of 7 = 15

$\Rightarrow P = (23 \times 15) \bmod 26$

$= (23 \bmod 26 \times 15 \bmod 26) \bmod 26$

$= 345 \bmod 26 = 7 \Rightarrow$ 'h'

Similarly :- $E = 04 \Rightarrow P = [(4-2) \times 7^{-1}] \bmod 26$

$\Rightarrow (2 \times 15) \bmod 26 = 30 \bmod 26$

$= 4 =$ 'e'

C: B → 01	Decryption: $((01 - 2) \times 7^{-1}) \bmod 26$	P:11 → l
C: B → 01	Decryption: $((01 - 2) \times 7^{-1}) \bmod 26$	P:11 → l
C: W → 22	Decryption: $((22 - 2) \times 7^{-1}) \bmod 26$	P:14 → o

Cryptanalysis

The size of the key space for the monoalphabetic substitution cipher is 26! (almost 4×10^{26}). This makes a brute-force attack extremely difficult for Eve even if she is using a powerful computer. However, she can use statistical attack based on the frequency of characters. The cipher does not change the frequency of characters.

Polyalphabetic Ciphers :-

01. Each occurrence of letter may have different substitution

ex 'A' beginning = 'D' end में 'A' = 'Z'

दोनों में 'A' = 'N'

02. Relation b/w a character in plain text to character in cipher text = one to many

03. Advantage :- Hiding the frequency of letters

To create a polyalphabetic cipher, we need to make each ciphertext character dependent on both the corresponding plaintext character and the position of the plaintext character in the message. This implies that our key should be a stream of subkeys, in which each subkey depends somehow on the position of the plaintext character that uses that subkey for encipherment. In other words, we need to have a key stream $k = (k_1, k_2, k_3, \ldots)$ in which k_i is used to encipher the ith character in the plaintext to create the ith character in the ciphertext.

In simpler terms, a polyalphabetic cipher works by using a key that changes with each letter of the message. Instead of using the same key for every letter (as in simpler ciphers), each letter in the message gets encrypted using a different part of the key.
Imagine you have a secret key that's made up of different smaller keys. Each small key (let's call them k1, k2, k3, etc.) is used to encrypt a specific letter in the message. For example, the first letter in the message will be encrypted using the first small key (k1), the second letter will be encrypted using the second small key (k2), and so on.
This changing key makes the encryption stronger, because even if two letters in the message are the same, they will be encrypted differently based on their position in the message and the corresponding key part used.

Autokey Cipher

To see the position dependency of the key, let us discuss a simple polyalphabetic cipher called the **autokey cipher.** In this cipher, the key is a stream of subkeys, in which each subkey is used to encrypt the corresponding character in the plaintext. The first subkey is a predetermined value secretly agreed upon by Alice and Bob. The second subkey is the value of the first plaintext character (between 0 and 25). The third subkey is the value of the second plaintext. And so on.

Explanation :-

Let's break down an example of an **Autokey Cipher** step by step:

Example:

- **Plaintext**: HELLO
- **Secret First Key (k1)**: B (which is the number 1, since A = 0, B = 1, C = 2, etc.)
- **Alphabet Positions**: A = 0, B = 1, C = 2, ..., Z = 25.

Steps:

1. **Step 1 (Encrypt the first letter):**

is a predetermined value secretly agreed upon by Alice and Bob. The second subkey is the value of the first plaintext character (between 0 and 25). The third subkey is the value of the second plaintext. And so on.

- **Plaintext**: HELLO
- **Secret First Key (k1)**: B (which is the number 1, since A = 0, B = 1, C = 2, etc.)
- **Alphabet Positions**: A = 0, B = 1, C = 2, ..., Z = 25.

Steps:

1. **Step 1 (Encrypt the first letter):**
   - Plaintext letter: **H** (Position 7 in the alphabet).
   - Subkey: **B** (Position 1).
   - Add positions: 7 + 1 = 8 (mod 26).
   - 8 corresponds to **I** in the alphabet.
   - So, the first ciphertext letter is **I**.
2. **Step 2 (Encrypt the second letter):**
   - Plaintext letter: **E** (Position 4 in the alphabet).
   - Subkey: **H** (Position 7, from the first plaintext letter).
   - Add positions: 4 + 7 = 11 (mod 26).
   - 11 corresponds to **L**.
   - So, the second ciphertext letter is **L**.
3. **Step 3 (Encrypt the third letter):**
   - Plaintext letter: **L** (Position 11 in the alphabet).
   - Subkey: **E** (Position 4, from the second plaintext letter).
   - Add positions: 11 + 4 = 15 (mod 26).
   - 15 corresponds to **P**.
   - So, the third ciphertext letter is **P**.
4. **Step 4 (Encrypt the fourth letter):**
   - Plaintext letter: **L** (Position 11).
   - Subkey: **L** (Position 11, from the third plaintext letter).
   - Add positions: 11 + 11 = 22 (mod 26).
   - 22 corresponds to **W**.
   - So, the fourth ciphertext letter is **W**.
5. **Step 5 (Encrypt the fifth letter):**
   - Plaintext letter: **O** (Position 14).
   - Subkey: **L** (Position 11, from the fourth plaintext letter).
   - Add positions: 14 + 11 = 25 (mod 26).
   - 25 corresponds to **Z**.
   - So, the fifth ciphertext letter is **Z**.

Final Ciphertext: ILPWZ

In summary:

- The first key was chosen (B = 1).
- Each subsequent key was derived from the corresponding plaintext letter.

01. first key is decided by the sender and reciever

02. The rest of the keys are just the letters of the plaintext.

03. step 2 में key = 'H' which is the first letter of plaintext.

Cryptanalysis :- Though the autokey cipher hides the freq. but it is still vulnerable to brute force attack bcoz the initial key is chosen from key domain(0-25) Key domain छोटा है।

Play fair Cipher

Secret Key =

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

let us encrypt → 'hello'

Step 01 :- break into number of pieces each piece containing pair of letters.

'he', 'lx', 'lo' → pairs.

Step 02

he	lx	lo
EC	BZ	BX

b and e ⇒ same row ⇒ consider right. H → E, E → C

lx = same column → l के नीचे वाला पकड़ो and x का भी

L → Q X → Z

lo ⇒ Not in same row and not in same column ⇒ form a rectangle ⇒ l के row का last B, e के row का first X

Cryptanalysis is very difficult as

hello = ECQZBX

Cryptanalysis is very difficult as it contains (25! keys)

Vigenere Cipher

One interesting kind of polyalphabetic cipher was designed by Blaise de Vigenere, a sixteenth-century French mathematician. A **Vigenere cipher** uses a different strategy to create the key stream. The key stream is a repetition of an initial secret key stream of length m, where we have $1 \le m \le 26$. The cipher can be described as follows where $(k_1, k_2, \ldots, k_m)$ is the initial secret key agreed to by Alice and Bob.

$P = P_1P_2P_3 \ldots \quad C = C_1C_2C_3 \ldots \quad K = [(k_1, k_2, \ldots, k_m), (k_1, k_2, \ldots, k_m), \ldots]$

Encryption: $C_i = P_i + k_i$ Decryption: $P_i = C_i - k_i$

One important difference between the Vigenere cipher and the other two polyalphabetic ciphers we have looked at, is that the Vigenere key stream does not depend on the plaintext characters; it depends only on the position of the character in the plaintext. In other words, the key stream can be created without knowing what the plaintext is.

Example 3.16

Let us see how we can encrypt the message "She is listening" using the 6-character keyword "*PASCAL*". The initial key stream is (15, 0, 18, 2, 0, 11). The key stream is the repetition of this initial key stream (as many times as needed).

Plaintext:	s	h	e	i	s	l	i	s	t	e	n	i	n	g
P's values:	18	07	04	08	18	11	08	18	19	04	13	08	13	06
Key stream:	***15***	***00***	***18***	***02***	***00***	***11***	***15***	***00***	***18***	***02***	***00***	***11***	***15***	***00***
C's values:	**07**	**07**	**22**	**10**	**18**	**22**	**23**	**18**	**11**	**6**	**13**	**19**	**02**	**06**
Ciphertext:	**H**	**H**	**W**	**K**	**S**	**W**	**X**	**S**	**L**	**G**	**N**	**T**	**C**	**G**

Hill Cipher

- To encrypt (decrypt) each block of m letters (m-component vector) is multiplied by an invertible ***m x m*** matrix (inverse of the matrix)

$$K = \begin{bmatrix} k_{11} & k_{12} & \ldots & k_{1m} \\ k_{21} & k_{22} & \ldots & k_{2m} \\ \vdots & \vdots & & \vdots \\ k_{m1} & k_{m2} & \ldots & k_{mm} \end{bmatrix}$$

$$C_1 = P_1 k_{11} + P_2 k_{21} + \cdots + P_m k_{m1}$$
$$C_2 = P_1 k_{12} + P_2 k_{22} + \cdots + P_m k_{m2}$$
$$\ldots$$
$$C_m = P_1 k_{1m} + P_2 k_{2m} + \cdots + P_m k_{mm}$$

The key matrix in the Hill cipher needs to have a multiplicative inverse.

Example:

Let's encrypt the message "HI" using a 2x2 matrix.

- Plaintext: HI
- Key Matrix:

$$\begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix}$$

- Step 1: Convert the Plaintext into Numbers:
  - H = 7, I = 8
  - So, the plaintext becomes the vector:

$$\begin{bmatrix} 7 \\ 8 \end{bmatrix}$$

- Step 2: Multiply the Key Matrix by the Plaintext Vector:

$$\begin{bmatrix} 3 & 3 \\ 2 & 5 \end{bmatrix} \times \begin{bmatrix} 7 \\ 8 \end{bmatrix} = \begin{bmatrix} (3 \times 7) + (3 \times 8) \\ (2 \times 7) + (5 \times 8) \end{bmatrix} = \begin{bmatrix} 45 \\ 54 \end{bmatrix}$$

  - Now, take the result mod 26 (since there are 26 letters in the alphabet):

$$\begin{bmatrix} 45 \mod 26 \\ 54 \mod 26 \end{bmatrix} = \begin{bmatrix} 19 \\ 2 \end{bmatrix}$$

- Step 3: Convert the Numbers Back into Letters:
  - 19 corresponds to **T** and 2 corresponds to **C**.

Cryptanalysis :-

01. very difficult b'coz matrix of key = m x m
can take any values from (0-25)
Hence key domain = $26^{m \times m}$

One time Pad :-

01. Key = same length of plaintext and key is chosen at random

02. Key is generated, used once and discarded

is chosen at random

02. Key is generated, used once and discarded

- Example: P = HELLO, Key = MONEY.
  - P = 7, 4, 11, 11, 4, K = 12, 14, 13, 4, 24.
  - C = P + K = 19, 18, 24, 15, 38 = 19, 18, 24, 15, 12 = TSYPM
  - P = C – K = 7, 4, 11, 11, -12, = 7, 4, 11, 11, 14 = HELLO

3.3 TRANSPOSITION CIPHERS

A transposition cipher does not substitute one symbol for another, instead it changes the location of the symbols. A symbol in the first position of the plaintext may appear in the tenth position of the ciphertext. A symbol in the eighth position in the plaintext may appear in the first position of the ciphertext. In other words, a transposition cipher reorders (transposes) the symbols.

A transposition cipher reorders symbols.

*Keyless Transposition:-

Simple transposition ciphers used in the past = keyless.

2 methods — ① Method-01:- text is written in table column by column and transmitted row by row

② Method 02:- written row by row transmitted column by column

m	e	m	a	t	e	a	k
e	t	e	t	h	p	r	

← example of method 01.

transmitted row by row

Keyed Transposition Ciphers

The keyless ciphers permute the characters by using writing plaintext in one way (row by row, for example) and reading it in another way (column by column, for example). The permutation is done on the whole plaintext to create the whole ciphertext. Another method is to divide the plaintext into groups of predetermined size, called blocks, and then use a key to permute the characters in each block separately.

e n e m y a t t a c k s t o n i g h t z

The key used for encryption and decryption is a permutation key, which shows how the character are permuted. For this message, assume that Alice and Bob used the following key:

Encryption ↓	3	1	4	5	2
	1	2	3	4	5

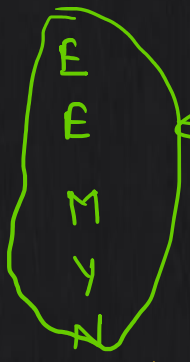
↑ Decryption

The third character of plaintext will be placed at first position of Ciphertext.

e is third character → I position

e first = II position

m fourth = III



← ciphers text.

Combining Two approaches:-

① Put it into table → row by row apply the keyed transformation read it column by column

① write column by column

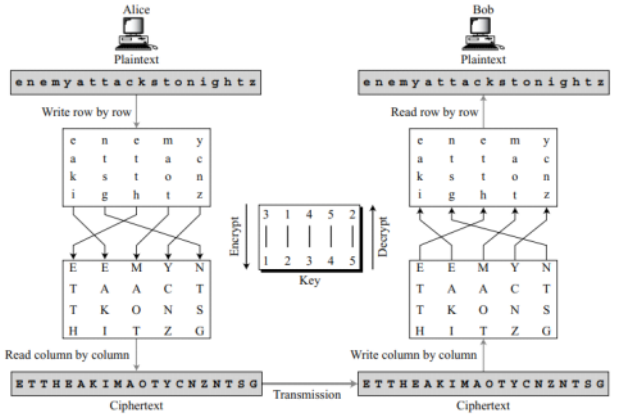
decrypt using key transformation

apply the key ... read it column by column and send

decrypt using key transformarm read it rowbyrow

Answer

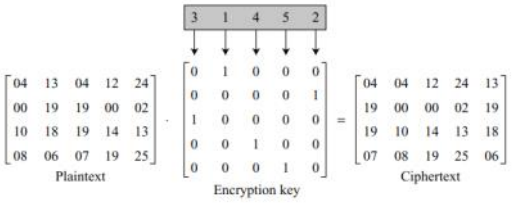
Figure 3.21 *Example 3.27*



Example 3.27

Figure 3.24 shows the encryption process. Multiplying the 4×5 plaintext matrix by the 5×5 encryption key gives the 4×5 ciphertext matrix. Matrix manipulation requires changing the characters in Example 3.27 to their numerical values (from 00 to 25). Note that the matrix multiplication provides only the column permutation of the transposition; reading and writing into the matrix should be provided by the rest of the algorithm.

Figure 3.24 *Representation of the key as a matrix in the transposition cipher*



Cryptanalysis:-

Statistical Attack

A transposition cipher does not change the frequency of letters in the ciphertext; it only reorders the letters. So the first attack that can be applied is single-letter frequency analysis. This method can be useful if the length of the ciphertext is long enough. We have seen this attack before. However, transposition ciphers do not preserve the frequency of digrams and trigrams. This means that Eve cannot use these tools. In fact, if a cipher does not preserve the frequency of digrams and trigrams, but does preserve the frequency of single letters, it is probable that the cipher is a transposition cipher.

Brute-Force Attack

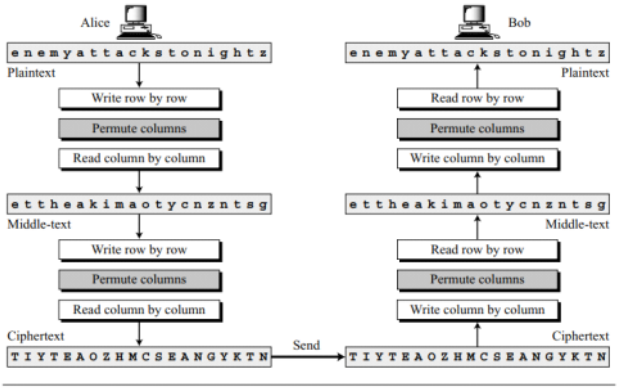
Eve can try all possible keys to decrypt the message. However, the number of keys can be huge ($1! + 2! + 3! + \cdots + L!$), where L is the length of the ciphertext. A better approach is to guess the number of columns. Eve knows that the number of columns divides L. For example, if the length of the cipher is 20 characters, then $20 = 1 \times 2 \times 2 \times 5$.

This means the number of columns can be a combination of these factors (1, 2, 4, 5, 10, 20). However, the first (only one column) is out of the question and the last (only one row) is unlikely.

Double Transposition Ciphers

Double transposition ciphers can make the job of the cryptanalyst difficult. An example of such a cipher would be the one that repeats twice the algorithm used for encryption and decryption in Example 3.26. A different key can be used in each step, but normally the same key is used.

Figure 3.25 *Double transposition cipher*



TIYTEAOZHMCSEANGYKTN → TIYTEAOZHMCSEANGYKTN

3.4 STREAM AND BLOCK CIPHERS

Symmetric Ciphers
- Stream Cipher
- Block Cipher

(A) Stream Cipher :- 01. Encryption and decryption are done one symbol at a time.

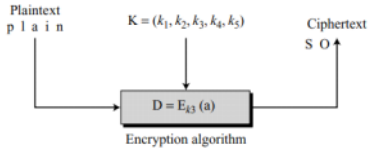
02.

$$P = P_1P_2P_3, \ldots \qquad C = C_1C_2C_3, \ldots \qquad K = (k_1, k_2, k_3, \ldots)$$
$$C_1 = E_{k1}(P_1) \qquad C_2 = E_{k2}(P_2) \qquad C_3 = E_{k3}(P_3) \ldots$$

We have three streams plaintext, ciphertext, keys

03 Characters of the plaintext are fed to the encryption algo. one at a time and the ciphertext characters are also created one at a time.

Figure 3.26 *Stream cipher*



Additive ciphers can be categorized as stream ciphers in which the key stream is the repeated value of the key. In other words, the key stream is considered as a predetermined stream of keys or $K = (k, k, \ldots, k)$. In this cipher, however, each character in the ciphertext depends only on the corresponding character in the plaintext, because the key stream is generated independently.

Example 3.31

The monoalphabetic substitution ciphers discussed in this chapter are also stream ciphers. However, each value of the key stream in this case is the mapping of the current plaintext character to the corresponding ciphertext character in the mapping table.

Example 3.32

Vigenere ciphers are also stream ciphers according to the definition. In this case, the key stream is a repetition of m values, where m is the size of the keyword. In other words,

$$K = (k_1, k_2, \ldots k_m, k_1, k_2, \ldots k_m, \ldots)$$

Block Cipher :-

In a **block cipher**, instead of encrypting each letter or symbol individually, you take a group of symbols (called a **block**) and encrypt them together as one unit. Each block has a fixed size, let's call it **m**, which means **m symbols** are grouped and encrypted at the same time.

Key Points:

1. **Group of Symbols**:
   - Imagine your message is divided into chunks of **m** symbols (where **m** is greater than 1). For example, if **m = 4**, you might group a message like "HELLOWORLD" into blocks like "HELL", "OWOR", and "LDXX" (with padding "XX" if needed to complete the last block).
2. **One Key for the Whole Block**:
   - You use a **single key** to encrypt each block. Even though the key might be made up of multiple values or numbers, the whole block is encrypted using that one key. This means the key doesn't change for each symbol in the block, but instead, it applies to the entire block at once.
3. **Same Size**:
   - The **output (ciphertext)** block is the same size as the **input (plaintext)** block. So, if you're encrypting a block of 4 symbols, the encrypted result will also have 4 symbols, but they'll be scrambled into something unrecognizable.

Example (Simplified):

If the block size **m = 4**, and you have a message like "HELP", you would encrypt the entire group of 4 letters together as one block, using the same key for the whole block. This produces an encrypted group (ciphertext) that might look like "XMAL", which still has 4 characters.

Summary:

- **Block cipher** means encrypting groups of symbols together as blocks, not one by one.
- A **single key** is used to encrypt each block, regardless of how many symbols it contains.
- The encrypted output (ciphertext) has the **same size** as the original block of symbols.

Every block cipher → polyalphabetic cipher