**1 Shift Cipher :**

The shift cipher (also called Caesar cipher) works as follows. The English alphabet is represented by numbers from 0 to 25 i.e., {A, B, . . . , Z} are mapped to {0, 1, . . . , 25} in the same order. Define K = {0, 1, 2, . . . , 25}, M = C = {0, 1, 2, . . . , 25}. Gen(): k U ← K Enc(k, m = m1m2 . . . mn): Set ci ← (mi + k) mod 26. Ciphertext is given by c = c1c2 . . . cn. Dec(k, c = c1c2 . . . cn): Recover message components as mi ← (ci − k) mod 26.

1. **Is this encryption perfectly secret?**

No ,the cipher text is not perfectly secret . An encryption means that it provides no information about the text. But in this case every character in the encrypted message is shifted by a fixed amount .It is a ciphertext , though there only 25 possible keys in Caesar cipher it can be decrypted by a brute force.

1. **What change can we make to the key so that it becomes perfectly secret? (Hint: Can increasing length of key help if we modify encryption scheme in some way?)**

Increasing the complexity of encryption keys by using a wide range of characters, including upper case and lower case characters ,numbers and special symbols. To enhance the protection against brute force attacks, it is advisable to use encryption keys that are lengthy and randomly generated .

Some advance methods for encryption even with infinite computational power , attacker cannot gain any information:

(1)One Time Pad(OTP)

(2)Quantum Key Distribution(QKD)

(3)Perfect Forward Secrecy(PFC)

(4)Key derivation from Biometrics

(5)Post Quantum Cryptography(PQC)

**2 Let’s code :**

Seeing that you all might be quite familiar with shift cipher now, lets try to implement this. Assume we have a key ranging from 0 to 25 and a lowercase English message. (You can assume that space is not encrypted)

(a) Write a function Enc(m, k) that takes 2 parameters m and k, the message and key respectively, to encrypt the message using shift cipher. And so find the encrypted versions of the following messages - (i) ’iitk is better than iitd and iitb’ with k = 9 (ii) ’lets learn cryptography’ with k = 25

(b) Given an encrypted message, write a function to list all possible original messages with a randomized key. Also given the original message was intelligible, find the most probable message from the list. (i) ’bm ptl wtfg xtlr tztbg’ (ii) ’rc fjb mjvw njbh’

(a) def Enc(m,k):

    encrypt=""

    for char in m:

          if char.islower():

               new\_char = chr(((ord(char) - 97 + k) % 26) + 97)

               encrypt+=new\_char

          elif char.isupper():

               new\_char= chr(((ord(char) - 65 + k) % 26) + 65)

               encrypt+=new\_char

          else:

               encrypt+=char

    return encrypt

m1 = 'iitk is better than iitd and iitb'

k1 = 9

encrypted\_message1 = Enc(m1, k1)

print("Encrypted message 1:", encrypted\_message1)

m2 = 'lets learn cryptography'

k2 = 25

encrypted\_message2 = Enc(m2, k2)

print("Encrypted message 2:", encrypted\_message2)

**OUTPUT:**

**Encrypted message 1: rrct rb knccna cqjw rrcm jwm rrck**

**Encrypted message 2: kdsr kdzqm bqxosnfqzogx**

(b) def Dec(m, k):

    decrypt= ''

    for char in m:

            if char.isalpha():

                if char.islower() :

                     new\_char = chr(((ord(char) - 97 - k) % 26) + 97)

                elif char.isupper():

                     chr(((ord(char) - 65 - k) % 26) + 65)

                decrypt += new\_char

            else:

                 decrypt+=char

    return decrypt

def possible\_messages(encrypt\_message):

    possible\_messages = []

    for key in range(1,26):

        possible\_messages.append(Dec(encrypt\_message, key))

    return possible\_messages

message1 = 'bm ptl wtfg xtlr tztbg'

message2 = 'rc fjb mjvw njbh'

print(" probable message 1:",possible\_messages(message1))

print(" probable message 2:",possible\_messages(message2))

**OUTPUT:**

**probable message 1: ['al osk vsef wskq sysaf', 'zk nrj urde vrjp rxrze', 'yj mqi tqcd uqio qwqyd', 'xi lph spbc tphn pvpxc', 'wh kog roab sogm ouowb', 'vg jnf qnza rnfl ntnva', 'uf ime pmyz qmek msmuz', 'te hld olxy pldj lrlty', 'sd gkc nkwx okci kqksx', 'rc fjb mjvw njbh jpjrw', 'qb eia liuv miag ioiqv', 'pa dhz khtu lhzf hnhpu', 'oz cgy jgst kgye gmgot', 'ny bfx ifrs jfxd flfns', 'mx aew heqr iewc ekemr', 'lw zdv gdpq hdvb djdlq', 'kv ycu fcop gcua cickp', 'ju xbt ebno fbtz bhbjo', 'it was damn easy again', 'hs vzr czlm dzrx zfzhm', 'gr uyq bykl cyqw yeygl', 'fq txp axjk bxpv xdxfk', 'ep swo zwij awou wcwej', 'do rvn yvhi zvnt vbvdi', 'cn qum xugh yums uauch']**

**probable message 2: ['qb eia liuv miag', 'pa dhz khtu lhzf', 'oz cgy jgst kgye', 'ny bfx ifrs jfxd', 'mx aew heqr iewc', 'lw zdv gdpq hdvb', 'kv ycu fcop gcua', 'ju xbt ebno fbtz', 'it was damn easy', 'hs vzr czlm dzrx', 'gr uyq bykl cyqw', 'fq txp axjk bxpv', 'ep swo zwij awou', 'do rvn yvhi zvnt', 'cn qum xugh yums', 'bm ptl wtfg xtlr', 'al osk vsef wskq', 'zk nrj urde vrjp', 'yj mqi tqcd uqio', 'xi lph spbc tphn', 'wh kog roab sogm', 'vg jnf qnza rnfl', 'uf ime pmyz qmek', 'te hld olxy pldj', 'sd gkc nkwx okci']**

Most intelligible for decrypted message 1 is: **'it was damn easy again'**

Most intelligible for decrypted message 2 is: **'it was damn easy '**