**DEPARTMENT OF INFORMATION TECHNOLOGY**

**COURSE CODE: DJ19ITL501 DATE: 4/10/22**

**COURSE NAME: Cryptography and Network Security Lab Class: A3**

**LAB EXPERIMENT NO. 1**

**AIM:** Design and Implementation of a product cipher using Substitution and Transposition ciphers

**THEORY:**

In cryptography, a product cipher combines two or more transformations in a manner intending that the resulting cipher is more secure than the individual components to make it resistant to cryptanalysis. By combining two or more simple [transposition ciphers](https://www.britannica.com/topic/transposition-cipher) or [substitution ciphers](https://www.britannica.com/topic/substitution-cipher), a more secure encryption may result.

**DESIGN AND IMPLEMENTATION CODE:**

1. **Substitution cipher(PLAYFAIR):**

#playfair substitution cipher

#p="hello world"

#print(p)

def find(element, matrix):

    for i, matrix\_i in enumerate(matrix):

        for j, value in enumerate(matrix\_i):

            if value == element:

                return (i, j)

def encrypty(p,key):

    p=p.replace(" ","")

    if(len(p)%2!=0):

        p=p+"z"

    #print(p)

    blocks=[]

    for i in range(2,len(p)+1,2):

        blocks.append(p[ i-2: i])

    cblocks=[]

    for i in range(0,len(blocks)):

        if(blocks[i][1]==blocks[i][0]):

            b1=blocks[i][0]+"z"

            b2=blocks[i][1]+"z"

            cblocks.append(b1)

            cblocks.append(b2)

        else:

            cblocks.append(blocks[i])

    c=""

    for i in cblocks:

            p1=find(i[0],key)

            p2=find(i[1],key)

            if p1[0]==p2[0]:

                if(p1[1]==4):

                    c=c+key[p1[0]][0]

                    c=c+key[p2[0]][p2[1]+1]

                elif(p2[1]==4):

                    c=c+key[p1[0]][p1[1]+1]

                    c=c+key[p2[0]][0]

                else:

                    c=c+key[p1[0]][p1[1]+1]

                    c=c+key[p2[0]][p2[1]+1]

            elif p1[1]==p2[1]:

                if(p1[0]==4):

                    c=c+key[0][p1[1]]

                    c=c+key[p2[0]][p2[1]]

                elif(p2[0]==4):

                    c=c+key[p1[0]][p1[1]]

                    c=c+key[0][p2[1]]

                else:

                    c=c+key[p1[0]+1][p1[1]]

                    c=c+key[p2[0]+1][p2[1]]

            else:

                c=c+key[p1[0]][p2[1]]

                c=c+key[p2[0]][p1[1]]

    #print(cblocks)

    #print(c)

    return c

    #print(cblocks)

def decrypty(p,key):

    print(p)

    blocks=[]

    for i in range(2,len(p)+1,2):

        blocks.append(p[ i-2: i])

    cblocks=blocks

    c=""

    for i in cblocks:

            p1=find(i[0],key)

            p2=find(i[1],key)

            if p1[0]==p2[0]:

                if(p1[1]==0):

                    c=c+key[p1[0]][4]

                    c=c+key[p2[0]][p2[1]-1]

                elif(p2[1]==0):

                    c=c+key[p1[0]][p1[1]-1]

                    c=c+key[p2[0]][4]

                else:

                    c=c+key[p1[0]][p1[1]-1]

                    c=c+key[p2[0]][p2[1]-1]

            elif p1[1]==p2[1]:

                if(p1[0]==0):

                    c=c+key[4][p1[1]]

                    c=c+key[p2[0]][p2[1]]

                elif(p2[0]==0):

                    c=c+key[p1[0]][p1[1]]

                    c=c+key[4][p2[1]]

                else:

                    c=c+key[p1[0]-1][p1[1]]

                    c=c+key[p2[0]-1][p2[1]]

            else:

                c=c+key[p1[0]][p2[1]]

                c=c+key[p2[0]][p1[1]]

    #print(cblocks)

    #print(c)

    return c

#k=[["r","p","m","l","d"],["s","a","x","i","c"],["h","k","q","u","y"],["e","w","o","z","g"],["b","f","t","v","n"]]

#encrypty(p,k)

#decrypty("ebivivzoemdr",k)

1. **Transposition cipher(COMBINING KEY COLUMN):**

# keyed transposition cipher

#key=[2,0,3,4,1]

#p="enemy attacks tonight"

def encrypty(p,key):

    p=p.replace(" ","")

    if(len(p)%len(key)!=0):

        while(len(p)%len(key)!=0):

            p=p+"z"

    #print(p)

    blocks=[]

    for i in range(5,len(p)+1,len(key)):

        blocks.append(p[ i-5: i])

    cwords=[]

    #print(blocks)

    for word in blocks:

        cword=""

        for i in key:

            cword=cword+word[i]

        cwords.append(cword)

    #print(cwords)

    c=""

    for i in range(0,len(key)):

        for word in cwords:

            c=c+word[i]

    return c

def decrypty(p,key):

    cols=len(p)//len(key)

    #print(len(key))

    blocks=[]

    for i in range(cols,len(p)+1,cols):

        blocks.append(p[ i-cols: i])

    cblocks=[]

    #print(blocks)

    for i in range(0,len(blocks)):

        cblocks.append(blocks[key.index(i)])

    c=""

    for i in range(0,cols):

        for word in cblocks:

            c=c+word[i]

    return c

#encrypty(p,key)

#encrypty("lhloerwldozhzzi",key)

#decrypty("isxapocsdswxbar",[3,0,2,4,1])

1. **Client:**

import socket

import exp1 as e

import exp1tras as t

k=[["r","p","m","l","d"],["s","a","x","i","c"],["h","k","q","u","y"],["e","w","o","z","g"],["b","f","t","v","n"]]

key=[2,0,3,4,1]

#encrypty(p,k)

def Main():

    host='192.168.111.1' #client ip

    port = 4005

    server = ('192.168.111.1', 4000)

    s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

    s.bind((host,port))

    message = input("-> ")

    m=e.encrypty(message,k)

    m=t.encrypty(m,key)

    s.sendto(m.encode('utf-8'), server)

    s.close()

if \_\_name\_\_=='\_\_main\_\_':

    Main()

1. **Server:**

import socket

import exp1 as e

import exp1tras as t

k=[["r","p","m","l","d"],["s","a","x","i","c"],["h","k","q","u","y"],["e","w","o","z","g"],["b","f","t","v","n"]]

key=[2,0,3,4,1]

#encrypty(p,k)

def Main():

    host = '192.168.111.1' #Server ip

    port = 4000

    s = socket.socket(socket.AF\_INET, socket.SOCK\_DGRAM)

    s.bind((host, port))

    print("Server Started")

    while True:

        data, addr = s.recvfrom(1024)

        data = data.decode('utf-8')

        data= t.decrypty(data,key)

        data= e.decrypty(data,k)

        print("Message from: " + str(addr))

        print("From connected user: " + data)

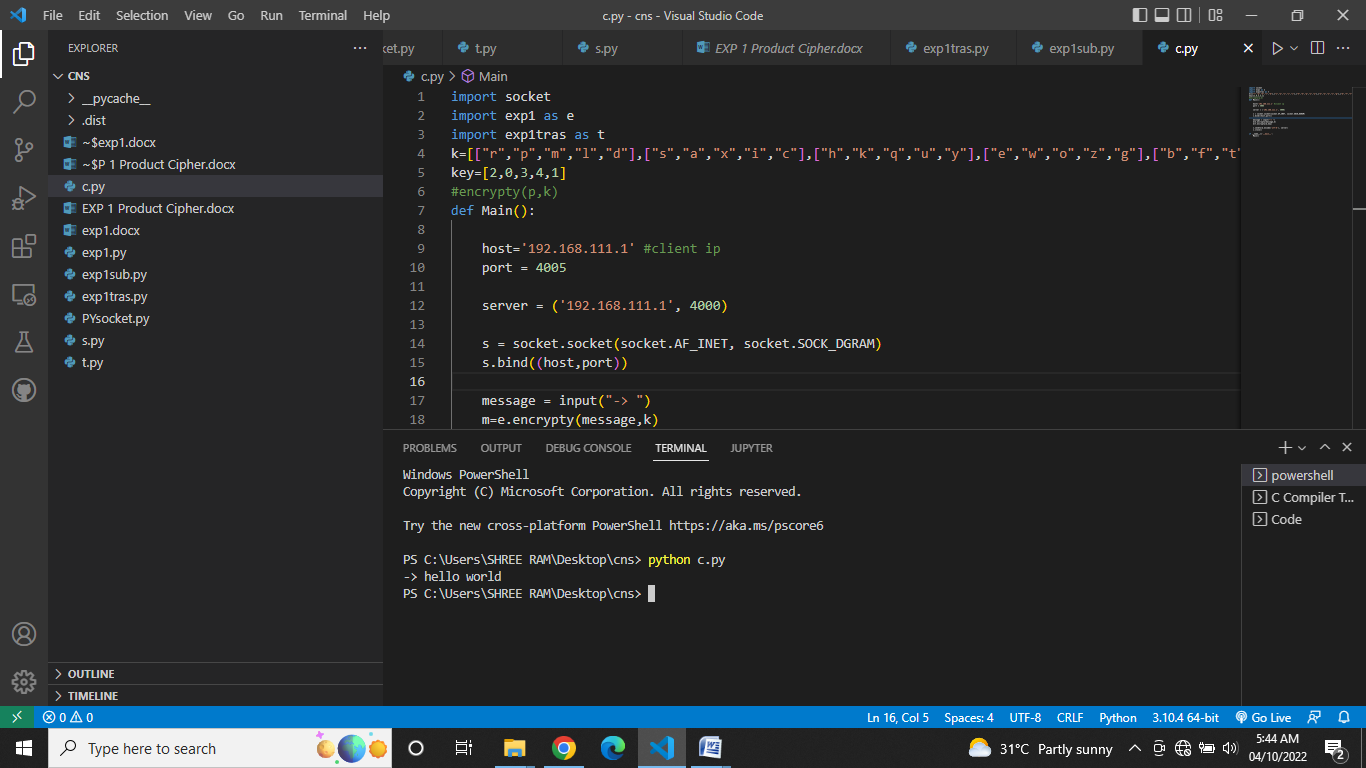
    c.close()

if \_\_name\_\_=='\_\_main\_\_':

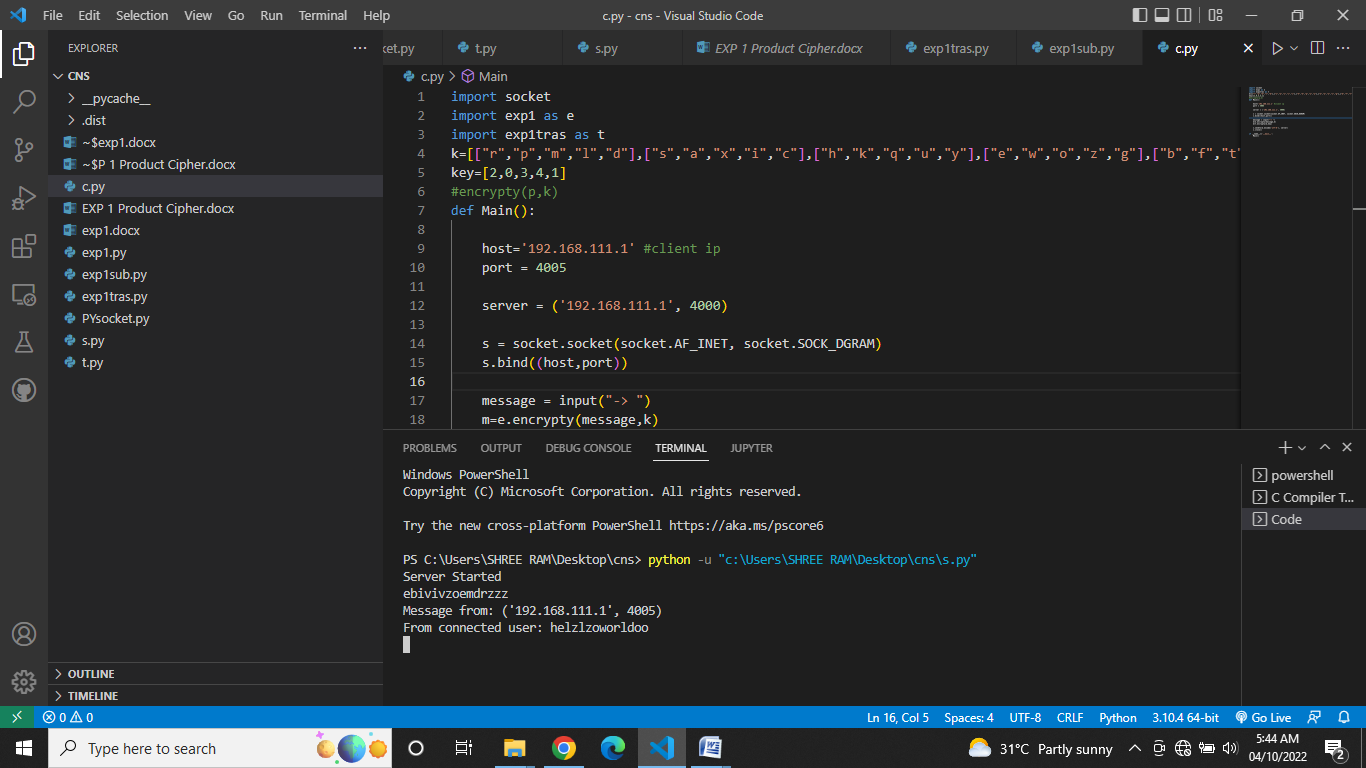
    Main()

**OUTPUT:**

**Client:**

****

**Server:**

****

**CONCLUSION:**

Product Ciphers improve security. Modern Ciphers use product ciphers. Thus we have designed and implemented a product cipher that is more secure against attacks such as brute force and statistical attacks.

**​**