Report on

CE708-7-AU– Computer Security

Lab Assignment 1: Ceasar Cipher and Playfair Cipher

Student Name : Atiwat Onsuwan (ao18409)

1. Decrypt the following message in Python. [3%]
   te td fyvyzhy szh pqqpnetgp esp nlpdlc ntaspc hld le esp etxp, mfe te td wtvpwj ez slgp mppy cpldzylmwj dpnfcp, yze wplde mpnlfdp xzde zq nlpdlc'd pypxtpd hzfwo slgp mppy twwtepclep lyo zespcd hzfwo slgp lddfxpo esle esp xpddlrpd hpcp hcteepy ty ly fyvyzhy qzcptry wlyrflrp.

Program

```
print('Encrypted msg is :')

ms = """te td fyvyzhy szh pqqpnetgp esp nlpdlc ntaspc hld le esp etxp, mfe te td
wtvpwj
ez slgp mppy cpldzylmwj dpnfcp, yze wplde mpnlfdp xzde zq nlpdlc'd pypxtpd
hzfwo slgp mppy twwtepclep lyo zespcd hzfwo slgp lddfxpo esle esp xpddlrpd
hpcp hcteepy ty ly fyvyzhy qzcptry wlyrflrp"""

print(ms)  # display the encrypted message from "ms"


# this function will plus number of shift to "text"(ms) depends on the parameter
"shift"
def decrypted_text(text, shift):
    a = ord('a')  # we use ord to
    return ''.join(
        chr((ord(char) - a + shift) % 26 + a) if 'a' <= char <= 'z' else char
        # get the new letter in 'ms' using ord at 'shift'
        for char in text.lower())  # turn every text in 'text' to lower case if it not
the lower case


for i in range(1, 100):  # set loop to 100 times
    if i < 100:
        ipt = input("Is this your plaintext ? y/n ")
        if ipt == "n" or ipt == "N":  # if user input n or N will do this condition
            print("=================================================================")
            print(decrypted_text(ms, i))  # send the message from 'ms' and the number
of shift that increases by 1 as 'i' every time through function 'decrypted_text'
        if ipt == "y" or ipt == "Y":  # if user input y or Y will do this condition
            i-=1
            print("=================================================================")
            print("cracked!!, number of shift(s) = ", i)  # display the number of
shift
            print("=================================================================")
            break  # break the loop
        else:
            print("Please enter y/n")

input(
```

Output

Encrypted msg is :

te td fyvyzhy szh pqqpnetgp esp nlpdlc ntaspc hld le esp etxp, mfe te td wtvpwj

ez slgp mppy cpldzylmwj dpnfcp, yze wplde mpnlfdp xzde zq nlpdlc'd pypxtpd

hzfwo slgp mppy twwtepclep lyo zespcd hzfwo slgp lddfxpo esle esp xpddlrpd

hpcp hcteepy ty ly fyvyzhy qzcptry wlyrflrp

Is this your plaintext ? y/n n

==========

uf ue gzwzaiz tai qrrqofuhq ftq omqemd oubtqd ime mf ftq fuyq, ngf uf ue xuwqxk

fa tmhq nqqz dqmeazmnxk eqogdq, zaf xqmef nqomgeq yaef ar omqemd'e qzqyuqe

iagxp tmhq nqqz uxxufqdmfq mzp aftqde iagxp tmhq meegyqp ftmf ftq yqeemsqe

iqdq iduffqz uz mz gzwzaiz radqusz xmzsgmsq

Please enter y/n

Is this your plaintext ? y/n n

==========

vg vf haxabja ubj rssrpgvir gur pnrfne pvcure jnf ng gur gvzr, ohg vg vf yvxryl

gb unir orra ernfbanoyl frpher, abg yrnfg orpnhfr zbfg bs pnrfne'f rarzvrf

jbhyq unir orra vyyvgrengr naq bguref jbhyq unir nffhzrq gung gur zrffntrf

jrer jevggra va na haxabja sbervta ynathntr

Please enter y/n

Is this your plaintext ? y/n n

==========

wh wg ibybckb vck sttsqhwjs hvs qosgof qwdvsf kog oh hvs hwas, pih wh wg zwyszm

hc vojs pssb fsogcbopzm gsqifs, bch zsogh psqoigs acgh ct qosgof'g sbsawsg

kcizr vojs pssb wzzwhsfohs obr chvsfg kcizr vojs oggiasr hvoh hvs asggousg

ksfs kfwhhsb wb ob ibybckb tcfswub zobuious

Please enter y/n

Is this your plaintext ? y/n n

==========

xi xh jczcdlc wdl tuutrixkt iwt rpthpg rxewtg lph pi iwt ixbt, qji xi xh axztan

id wpkt qttc gtphdcpqan htrjgt, cdi atphi qtrpjht bdhi du rpthpg'h tctbxth

ldjas wpkt qttc xaaxitgpit pcs diwtgh ldjas wpkt phhjbts iwpi iwt bthhpvth

ltgt lgxiitc xc pc jczcdlc udgtxvc apcvjpvt

Please enter y/n

Is this your plaintext ? y/n n

==========

yj yi kdademd xem uvvusjylu jxu squiqh syfxuh mqi qj jxu jycu, rkj yj yi byaubo

je xqlu ruud huqiedqrbo iuskhu, dej buqij rusqkiu ceij ev squiqh'i uducyui

mekbt xqlu ruud ybbyjuhqju qdt ejxuhi mekbt xqlu qiikcut jxqj jxu cuiiqwui

muhu mhyjjud yd qd kdademd vehuywd bqdwkqwu

Please enter y/n

Is this your plaintext ? y/n n

==========

zk zj lebefne yfn vwwvtkzmv kyv trvjri tzgyvi nrj rk kyv kzdv, slk zk zj czbvcp

kf yrmv svve ivrjferscp jvtliv, efk cvrjk svtrljv dfjk fw trvjri'j vevdzvj

nflcu yrmv svve zcczkvirkv reu fkyvij nflcu yrmv rjjldvu kyrk kyv dvjjrxvj

nviv nizkkve ze re lebefne wfivzxe crexlrxv

Please enter y/n

Is this your plaintext ? y/n n

==========

al ak mfcfgof zgo wxxwulanw lzw uswksj uahzwj osk sl lzw laew, tml al ak dacwdq

lg zsnw twwf jwskgfstdq kwumjw, fgl dwskl twusmkw egkl gx uswksj'k wfweawk

ogmdv zsnw twwf addalwjslw sfv glzwjk ogmdv zsnw skkmewv lzsl lzw ewkksywk

owjw ojallwf af sf mfcfgof xgjwayf dsfymsyw

Please enter y/n

Is this your plaintext ? y/n n

==========

bm bl ngdghpg ahp xyyxvmbox max vtxltk vbiaxk ptl tm max mbfx, unm bm bl ebdxer

mh atox uxxg kxtlhgtuer lxvnkx, ghm extlm uxvtnlx fhlm hy vtxltk'l xgxfbxl

phnew atox uxxg beebmxktmx tgw hmaxkl phnew atox tllnfxw matm max fxlltzxl

pxkx pkbmmxg bg tg ngdghpg yhkxbzg etgzntzx

Please enter y/n

Is this your plaintext ? y/n n

==========

cn cm ohehiqh biq yzzywncpy nby wuymul wcjbyl qum un nby ncgy, von cn cm fceyfs

ni bupy vyyh lyumihuvfs mywoly, hin fyumn vywuomy gimn iz wuymul'm yhygcym

qiofx bupy vyyh cffcnyluny uhx inbylm qiofx bupy ummogyx nbun nby gymmuaym

qyly qlcnnyh ch uh ohehiqh zilycah fuhaouay

Please enter y/n

Is this your plaintext ? y/n n

==========

do dn pifijri cjr zaazxodqz ocz xvznvm xdkczm rvn vo ocz odhz, wpo do dn gdfzgt

oj cvqz wzzi mzvnjivwgt nzxpmz, ijo gzvno wzxvpnz hjno ja xvznvm'n zizhdzn

rjpgy cvqz wzzi dggdozmvoz viy joczmn rjpgy cvqz vnnphzy ocvo ocz hznnvbzn

rzmz rmdoozi di vi pifijri ajmzdbi gvibpvbz

Please enter y/n

Is this your plaintext ? y/n n

==========

ep eo qjgjksj dks abbaypera pda ywaown yeldan swo wp pda peia, xqp ep eo hegahu

pk dwra xaaj nawokjwxhu oayqna, jkp hawop xaywqoa ikop kb ywaown'o ajaieao

skqhz dwra xaaj ehhepanwpa wjz kpdano skqhz dwra wooqiaz pdwp pda iaoowcao

sana sneppaj ej wj qjgjksj bknaecj hwjcqwca

Please enter y/n

Is this your plaintext ? y/n n

==========

fq fp rkhkltk elt bccbzqfsb qeb zxbpxo zfmebo txp xq qeb qfjb, yrq fq fp ifhbiv

ql exsb ybbk obxplkxyiv pbzrob, klq ibxpq ybzxrpb jlpq lc zxbpxo'p bkbjfbp

tlria exsb ybbk fiifqboxqb xka lqebop tlria exsb xpprjba qexq qeb jbppxdbp

tbob tofqqbk fk xk rkhkltk clobfdk ixkdrxdb

Please enter y/n

Is this your plaintext ? y/n n

==========

gr gq slilmul fmu cddcargtc rfc aycqyp agnfcp uyq yr rfc rgkc, zsr gr gq jgicjw

rm fytc zccl pcyqmlyzjw qcaspc, lmr jcyqr zcaysqc kmqr md aycqyp'q clckgcq

umsjb fytc zccl gjjgrcpyrc ylb mrfcpq umsjb fytc yqqskcb rfyr rfc kcqqyecq

ucpc upgrrcl gl yl slilmul dmpcgel jylesyec

Please enter y/n

Is this your plaintext ? y/n n

==========

hs hr tmjmnvm gnv deedbshud sgd bzdrzq bhogdq vzr zs sgd shld, ats hs hr khjdkx

sn gzud addm qdzrnmzakx rdbtqd, mns kdzrs adbztrd lnrs ne bzdrzq'r dmdlhdr

vntkc gzud addm hkkhsdqzsd zmc nsgdqr vntkc gzud zrrtldc sgzs sgd ldrrzfdr

vdqd vqhssdm hm zm tmjmnvm enqdhfm kzmftzfd

Please enter y/n

Is this your plaintext ? y/n n

==========

it is unknown how effective the caesar cipher was at the time, but it is likely

to have been reasonably secure, not least because most of caesar's enemies

would have been illiterate and others would have assumed that the messages

were written in an unknown foreign language

Please enter y/n

Is this your plaintext ? y/n y

==========

cracked!!, number of shift(s) = 15

==========

2. Write a program in Python to: [7%]
   • read a plaintext and a key (user enters his preferred plaintext and the key) and encrypt the plaintext using Playfair Cipher.
   • read a Playfair Ciphertext and a key (user enters the Ciphertext and the key) and decrypt the Ciphertext.

Program

```
import re

letters_arr = []  # set 'letters_arr' as array
pin = 0  # set 'pin' = 0
for i in range(65, 91):  # loop to add every capital letter into 'letters_arr'
    letters_arr.append(chr(i)) # add value of chr ar i to letters_arr

k = 0  # set k=0 to use k to point the index number
table = [[0 for i in range(5)] for j in range(5)]  # create the matrix 5x5
for i in range(0, 5):  # looping the matrix to add every letter from 'letters_arr'
    for j in range(0, 5):
        table[i][j] = letters_arr[
            k]  # adding each letter in to matrix using 'i' and 'j' to point the index
to add store letters from 'letters_arr' at 'k'
        k += 1  # increase k by 1 every round of the looping


def gen_key_matrix(key):  # This funtion creates the matrix 5x5 using input key from
user
    alphabet = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'  # define possible letters that can be in
the key table after the user key is added
    gen_key_matrix.table = [[0] * 5 for row in range(5)]  # define the
'gen_key_matrix.table' as 2D array 5x5
    key = re.sub(r'[\W]', '',
                 key.upper())  # read string from 'key' and turn them into UPPERCASE
and pass new values to 'key'

    for row in range(5):  # outer loop at 5 times as 'row'
        for col in range(5):  # inner loop at 5 times as 'col'
            if len(key):  # do when length in key is TRUE
                gen_key_matrix.table[row][col] = key[
                    0]  # pass the value from 'key' at first index to
gen_key_matrix.table at index row and col
                alphabet = alphabet.replace(key[0], '')  # remove the value in
alphabet = key at first index
                key = key.replace(key[0], '', -1)  # replace '' at first index in key
and length -1
            else:  # do this when no value left in 'key'
                gen_key_matrix.table[row][col] = alphabet[
                    0]  # pass the value from alphabet at first index to
gen_key_matrix.table at index row and col
                alphabet = alphabet[1:]  # shift index of value in alphabet to the
next one
    return gen_key_matrix.table  # return value to gen_key_matrix.table


def position(table, letter): # loop from 1 to 25 position to find the position of
letter in table to compare with key table
    for row in range(5):
        for col in range(5):
            if table[row][col] == letter: # if found the position of letter return
```

```
value of row and col
                return [row, col]
    return [row, col]


def encrypt(): # this function encrypt the input plaintext base on input key
    texts = str(input("Enter plaintext:")) # get user plaintext input as string
    gen_key_matrix(key)
    table = gen_key_matrix.table # define 'table' = 'gen_key_matrix.table'
    cipher = '' # set cipher = ''
    texts = re.sub(r'[\W]', '', texts.upper()) # read string from 'texts' and turn
them into UPPERCASE and pass new values to 'key'
    text = '' # set text = ''

    for i in range(0, len(texts) - 1): # loop for length of 'texts' (number of
plaintext without J)
        text += texts[i] # add letters to'text' at 'texts' index i
        if texts[i] == texts[i + 1]: # if value in text at i = value in text at i+1
(same letter occur in the same pair) then replace the letter at i+1 with X
            text += 'X'

    text += texts[i + 1] #get pair by take the next letter of plaintext



    for i in range(0, len(text), 2):
        pair = text[i:i + 2] #take the letters from 'text' at i and i+2 to 'pair'
        a, b = pair[0], 'X' # a = text at index i, b = 'X'
        if len(pair) > 1: # if 'pair' > 1 or has pairs
            b = pair[1]  # then 'b' from 'X' = that letter
        a = position(table, a) # call function position to find the position of 'a' in
the key table
        b = position(table, b) # call function position to find the position of 'b' in
the key table

        if (a[0] == b[0]): # if 'a' and 'b' are in same row
            if a[1] == 4 and b[1] != 4: # if letter in 'a' is at the back but b is not
                cipher += table[a[0]][(a[1] - 4)] + table[b[0]][(b[1] + 1)] # cipher
will store letter at first index of its row
            elif b[1] == 4 and a[1] != 4: # if letter in 'b' is at the back but a is
not
                cipher += table[a[0]][(a[1] + 1)] + table[b[0]][(b[1] - 4)] # cipher
will store letter at first index of its row
            else: # if not match to conditions above but there are in same row
                cipher += table[a[0]][(a[1] + 1)] + table[b[0]][(b[1] + 1)] # cipher
will store letter from 'a' and 'b' at their positions + 1
        elif (a[1] == b[1]):# if 'a' and 'b' are in same column
            if a[0] == 4 and b[0] != 4: # if letter in 'a' is at the bottom but b is
not
                cipher += table[(a[0] - 4)][a[1]] + table[(b[0] + 1)][b[1]] # cipher
will store letter at top of its column
            elif b[0] == 4 and a[0] != 4:# if letter in 'b' is at the bottom but b is
not
                cipher += table[(a[0] + 1)][a[1]] + table[(b[0] - 4)][b[1]] # cipher
will store letter at top of its column
            else:# if not match to conditions above but there are in same column
                cipher += table[(a[0] + 1)][a[1]] + table[(b[0] + 1)][b[1]]  # cipher
will store letter from 'a' and 'b' at their positions + 1
        else: # if not == to any condition above mean that pair of plaintext is
diagonal
            cipher += table[a[0]][b[1]] + table[b[0]][a[1]] #'cipher' will store
letter at 'a' row and 'b' column, 'cipher' will store letter at 'b' row but 'a' column
    return cipher;
```

```
# this fuction is reverse way of encryption but I reduced condition by using "mod 5"
ex. if 'a'=0 and 'b'=1 are in same row and 'a' is at the front so a=(0-1)mod5 == 4,
b=(1-1)mod5 == 0
def decrypt(): # this function decrypt the input encrypted message base on input key
    text = str(input("Enter decrypted message:")) # get input decrypted message from
user
    gen_key_matrix(key)
    table = gen_key_matrix.table
    text = re.sub(r'[\W]', '', text.upper())
    texts = ''

    for i in range(0, len(text), 2):
        pair = text[i:i + 2]
        if len(pair) != 2:
            print('Please enter decrypted message')
            quit(-1)
        a, b = pair[0], pair[1]
        a = position(table, a)
        b = position(table, b)

        if a[0] == b[0]:
            texts += table[a[0]][(a[1] - 1) % 5] + table[b[0]][(b[1] - 1) % 5]
        elif (a[1] == b[1]):
            texts += table[(a[0] - 1) % 5][a[1]] + table[(b[0] - 1) % 5][b[1]]
        else:
            texts += table[a[0]][b[1]] + table[b[0]][a[1]]
    return texts


key = input("Enter your key :")  # getting user input into 'key'
key = key.replace(" ", "")  # remove space from 'key'
key = key.upper()  # turn letter in to UPPERCASE
gen_key_matrix(key)
ans = True  # set 'ans' as True
while ans:  # infinity loop to display selection menu using while loop
    print("""==Select Menu==
1.Encryption
2.Decryption
3.EXIT""")
    ans = input("Enter choice :")
    if ans == "1":
        message = encrypt()  # message = call function encrypt
        print("Your encrypted message is :", message,
              "\n")  # display encrypted message and return value of 'message' when
ans = "1"
    elif ans == "2":
        message = decrypt()  # message = call function encrypt
        print("Your plaintext is :", message, "\n")  # display plaintext and return
value of 'message' when ans = "2"
    elif ans == "3":
        break  # break the loop when ans = "3"
```

Output

```
Enter your key :tutorials
==Select Menu==
1.Encryption
2.Decryption
3.EXIT
Enter choice :1
Enter plaintext:hide money

Your encrypted message is : PCEFWSKGQY

==Select Menu==
1.Encryption
2.Decryption
3.EXIT
Enter choice :2
Enter decrypted message:PCEFWSKGQY
Your plaintext is : HIDEMONEYX

==Select Menu==
1.Encryption
2.Decryption
3.EXIT
Enter choice :1
Enter plaintext:juice

Your encrypted message is : KTCHGV

==Select Menu==
1.Encryption
2.Decryption
3.EXIT
Enter choice :2
Enter decrypted message:KTCHGV
Your plaintext is : JUICEX

==Select Menu==
1.Encryption
2.Decryption
3.EXIT
Enter choice :3
```