Assignment 2

Graded

Group

ANANYAE KUMAR BHARTARI
ABHAYA PRATAP SINGH
NAVEEN KUMAR MATHUR
View or edit group

Total Points

63 / 65 pts

Question 1

Team name

0 / 0 pts

✔ **+ 0 pts** Correct

+ 0 pts Incorrect

Question 2

Commands

10 / 10 pts

✔ **+ 10 pts** Correct

+ 0 pts Incorrect

Question 3

Cryptosystem

10 / 10 pts

✔ **+ 10 pts** Correct

+ 0 pts Incorrect

Question 4

Analysis

20 / 20 pts

✔ **+ 10 pts** Finding the key "SECURITY" from the Morse code

✔ **+ 10 pts** Figuring out the cryptosystem "PLAYFAIR"

+ 0 pts Wrong answer or NA

Question 5

Decryption algorithm

13 / 15 pts

✔ **+ 4 pts** How to create the 5×5 square matrix from the key and rest of the alphabets.

✔ **+ 8 pts** Decryption - Handling the extra "X" (2 marks). Handling the three cases while decrypting bigrams (3 x 2 marks)

✔ **+ 3 pts** Final decrypted text.

+ 0 pts Wrong answer or NA.

– 2 pts how are the extra "X"s handled?

Question 6

Password

10 / 10 pts

✔ **+ 10 pts** Correct

+ 0 pts Incorrect

Question 7

Code

0 / 0 pts

✔ **+ 0 pts** Correct

Q1 Team name
0 Points

NAA

Q2 Commands
10 Points

List the commands used in the game to reach the ciphertext.

go , back ,read

Q3 Cryptosystem
10 Points

What cryptosystem was used in this level?

play fair (digraph substitution cipher)

Q4 Analysis
20 Points

What tools and observations were used to figure out the cryptosystem? (Explain in less than 100 words)

On use of go command, we were able to access a screen which had the word "security" written in morse code(used morse table to decipher it). We noticed that "PLAY FAIR" was written in capital letters. We at first thought that it was a simple permutation cypher with block size 18 but no help. then we did a google search on types of permutation cyphers. there this "PLAY FAIR" was mentioned as example of substitution cypher. So we made a code and run the play fair cypher using python3 and found out the answer.

Q5 Decryption algorithm

15 Points

Briefly describe the decryption algorithm used. Also, mention the plaintext you deciphered. (Use less than 250 words)

The encrypting algorithm used here is PLAYFAIR algorithm. The exciting part of play fair algorithm is that the key remains the same on encryption or decryption.
The key square generated here is simply by a 5x5 matrix (total 25 entries) every letter in the matrix is distinct. there are 25 letters so we usually exclude "J" from the main text and replace it with "I". We first enter the different letters of the key into matrix then rest of the alphabets of English in alphabetic order that does not occur in the key. In this case, we used security to generate the key square. Hence the key square is -
| S E C U R |
| I T Y A B |
| D F G H K |
| L M N O P |
| Q V W X Z |
divide the crypt text into sequence of two letter called digraph now we substitute the two letters in following way:
1) If they lie in the same column then take the letters that are above each one. eg TM->EF. If it is at top, then going to bottom.
2)If they lie in the same row then take the letters to the left of each one. eg CR->EU. If is at leftmost, then going to rightmost.
3) adjacent letters like GH->FG and AH->UA by the above rules only
4)if they are not in a row or column then make a box with them at the corner and take the one at the opposite end(in the row). eg TR->BE

The plain text we deciphered is -
BEWARYOFTHENEXTCHAMBERTHEREISVERYLITTLEIOYTHERESPEAKOUTXTHEPASSWORDOPENSESAMETOGOTHROUGHMAYXYOUHAVETHESTRENGTHFORTHENEXTCHAMBERTOFINDTHEEXITYOUFIRSTWILXLNEXEDTOUTTERMAGICWORDSTHERE

Q6 Password

10 Points

What was the final command used to clear this level?

OPEN_SESAME

Q7 Code
0 Points

Upload any code that you have used to solve this level.

assignment_2.0 (1).py Download

```
import numpy as np
k="security"
k=k.lower()
l=len(k)
a="TR XYCB MH AFC MUVY EOHPTCS AFCSS TE QCSI NTYIMS TNA AFCSC EMRBH XAA VAFR MIUCQPUH LMRL_CCETOT FN HM AKUXAHK OTA WANAOTXT FFU EISCWNAF HME BFU MCVA UGTOTRE BM HYLF IFU UVTY ANEHBSEI QYOQM OUVSF AM EAFTE PYHYS XNSKE IFUSC"
# a=list(map(str,input().split()))
a=a.lower()
a=a.replace("j","i")
a=list(a)
an=list()
for i in a:
    if(i!=" " and i!="_"):
        an.append(i)

le=len(an)

if(le%2!=0):
    an.append("z")
    le+=1


b=list()
e=list()
for i in range(97,123):
    if(chr(i)!="j"):
        e.append(chr(i))
m=0
n=0
d={}
for i in range(5):
    c=[]
    while(len(c)!=5):
        if(n!=l and k[n] not in d):
            c.append(k[n])
            d[k[n]]=1
            n+=1
        elif(e[m] not in d):
            c.append(e[m])
            d[e[m]]=1
            m+=1
        else:
            m+=1

```

```
    b.append(c)


key=np.array(b)

r=np.where(key=="r")

res=[]

for i in range(0,le,2):
    r1=np.argwhere(key==an[i])
    r2=np.argwhere(key==an[i+1])

    if(r1[0][1] == r2[0][1]):
        if(r1[0][0]!=0):
            res.append(key[r1[0][0]-1][r1[0][1]])
        elif(r1[0][0]==0):
            res.append(key[4][r1[0][1]])

        if(r2[0][0]!=0):
            res.append(key[r2[0][0]-1][r2[0][1]])
        elif(r2[0][0]==0):
            res.append(key[4][r2[0][1]])
    elif(r1[0][0]==r2[0][0]):
        if(r1[0][1]!=0):
            res.append(key[r1[0][0]][r1[0][1]-1])
        elif(r1[0][1]==0):
            res.append(key[r1[0][0]][4])
        if(r2[0][1]!=0):
            res.append(key[r2[0][0]][r2[0][1]-1])
        elif(r2[0][1]==0):
            res.append(key[r2[0][0]][4])

    else:
        res.append(key[r1[0][0]][r2[0][1]])
        res.append(key[r2[0][0]][r1[0][1]])

final_string=str()
for i in range(len(res)):
    final_string+=res[i]
print(final_string)
```