Shri Vile Parle Kelavani Mandal's

DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING

(Autonomous College Affiliated to the University of Mumbai)

NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)

SUB: Information Security

AY 2023-24 (Semester-V)

Jhanvi Parekh
60009210033
D11

Experiment No: 4

Aim: Design and implement Encryption and Decryption Algorithm using Play fair Cipher.

Theory:

1. Playfair Cipher

Example:

1) Plaintext: ATTACK
Keyword: MONARCHY

Conclusion: The code showcases the encryption process of the Playfair cipher, a classical symmetric encryption technique that operates on pairs of letters. It illustrates how to prepare the plaintext by converting it to lowercase, removing spaces, and creating letter pairs (digraphs). Additionally, it handles cases where an odd number of letters result in the addition of a filler letter ('x') to ensure pairs.

The code proceeds to generate a key matrix (key table) based on a provided key and defines rules for encrypting digraphs within the matrix. Three rules—row, column, and rectangle rules—are implemented to determine the cipher text for each digraph. The resulting cipher text is displayed as the final output.

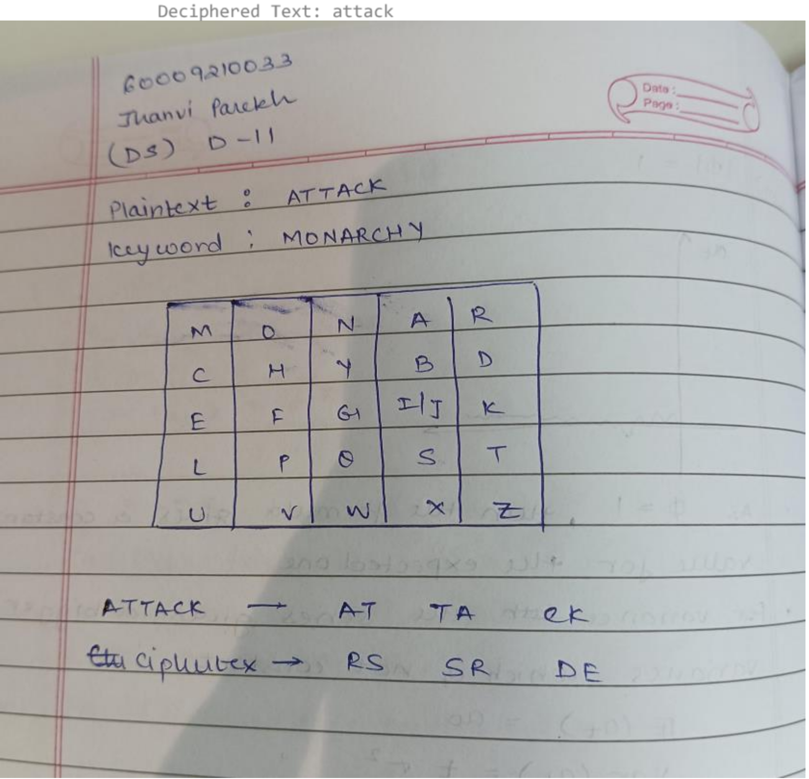
LINK:
https://colab.research.google.com/drive/1sCwuMiSDzdR923hlvyHaz5mVveUWfEBN?usp=sharing

```
def toLowerCase(text):
  return text.lower()
def removeSpaces(text):
  newText = ""
  for i in text:
    if i == " ":
      continue
    else:
      newText = newText + i
  return newText
def Diagraph(text):
  Diagraph = []
  group = 0
  for i in range(2, len(text), 2):
    Diagraph.append(text[group:i])
    group = i
  Diagraph.append(text[group:])
  return Diagraph

def FillerLetter(text):
  k = len(text)
  if k % 2 == 0:
    for i in range(0, k, 2):
      if text[i] == text[i+1]:
        new_word = text[0:i+1] + str('x') + text[i+1:]
        new_word = FillerLetter(new_word)
        break
      else:
        new_word = text
  else:
    for i in range(0, k-1, 2):
      if text[i] == text[i+1]:
        new_word = text[0:i+1] + str('x') + text[i+1:]
        new_word = FillerLetter(new_word)
        break
      else:
        new_word = text
  return new_word
list1 = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k', 'l', 'm',

    'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']

def generateKeyTable(word, list1):
  key_letters = []
  for i in word:
    if i not in key_letters:
      key_letters.append(i)
  compElements = []
  for i in key_letters:
    if i not in compElements:
      compElements.append(i)
  for i in list1:
    if i not in compElements:
      compElements.append(i)
  matrix = []
  while compElements != []:
    matrix.append(compElements[:5])
    compElements = compElements[5:]
  return matrix
def search(mat, element):
  for i in range(5):
    for j in range(5):
      if(mat[i][j] == element):
        return i, j

def encrypt_RowRule(matr, e1r, e1c, e2r, e2c):
  char1 = ''
  if e1c == 4:
    char1 = matr[e1r][0]
  else:
    char1 = matr[e1r][e1c+1]
  char2 = ''
  if e2c == 4:
    char2 = matr[e2r][0]
  return char1, char2

def encrypt_ColumnRule(matr, e1r, e1c, e2r, e2c):
  char1 = ''
  if e1r == 4:
    char1 = matr[0][e1c]
  else:
    char1 = matr[e1r+1][e1c]
  char2 = ''
  if e2r == 4:
    char2 = matr[0][e2c]
  else:
    char2 = matr[e2r+1][e2c]
  return char1, char2
def encrypt_RectangleRule(matr, e1r, e1c, e2r, e2c):
  char1 = ''
  char1 = matr[e1r][e2c]
  char2 = ''
  char2 = matr[e2r][e1c]

  return char1, char2
def encryptByPlayfairCipher(Matrix, plainList):
  CipherText = []
  for i in range(0, len(plainList)):
    c1 = 0
    c2 = 0
    ele1_x, ele1_y = search(Matrix, plainList[i][0])
    ele2_x, ele2_y = search(Matrix, plainList[i][1])
    if ele1_x == ele2_x:
      c1, c2 = encrypt_RowRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
    elif ele1_y == ele2_y:
      c1, c2 = encrypt_ColumnRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
    else:
      c1, c2 = encrypt_RectangleRule(
        Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
    cipher = c1 + c2
    CipherText.append(cipher)
  return CipherText
text_Plain = 'ATTACK'
text_Plain = removeSpaces(toLowerCase(text_Plain))
PlainTextList = Diagraph(FillerLetter(text_Plain))
if len(PlainTextList[-1]) != 2:
  PlainTextList[-1] = PlainTextList[-1]+'z'

key = "MONARCHY"
print("Key text:", key)
key = toLowerCase(key)
Matrix = generateKeyTable(key, list1)

print("Plain Text:", text_Plain)
CipherList = encryptByPlayfairCipher(Matrix, PlainTextList)

CipherText = ""
for i in CipherList:
  CipherText += i
print("CipherText:", CipherText)
```

```
Key text: MONARCHY
Plain Text: attack
CipherText: rssrde
```

```
def decryptByPlayfairCipher(Matrix, cipherList):
    DecipherText = []
    for i in range(0, len(cipherList)):
        c1 = 0
        c2 = 0
        ele1_x, ele1_y = search(Matrix, cipherList[i][0])
        ele2_x, ele2_y = search(Matrix, cipherList[i][1])

        if ele1_x == ele2_x:
            c1, c2 = decrypt_RowRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
        elif ele1_y == ele2_y:
            c1, c2 = decrypt_ColumnRule(Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
        else:
            c1, c2 = decrypt_RectangleRule(
                Matrix, ele1_x, ele1_y, ele2_x, ele2_y)
        DecipherText.append(decipher)
    return DecipherText
def decrypt_RowRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1c == 0:
        char1 = matr[e1r][4]
    else:
        char1 = matr[e1r][e1c - 1]
    char2 = ''
    if e2c == 0:
        char2 = matr[e2r][4]
    else:
        char2 = matr[e2r][e2c - 1]
    return char1, char2
def decrypt_ColumnRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    if e1r == 0:
        char1 = matr[4][e1c]
    else:
        char1 = matr[e1r - 1][e1c]
    char2 = ''
    if e2r == 0:
        char2 = matr[4][e2c]
    else:
        char2 = matr[e2r - 1][e2c]
    return char1, char2
def decrypt_RectangleRule(matr, e1r, e1c, e2r, e2c):
    char1 = ''
    char1 = matr[e1r][e2c]
    char2 = ''
    char2 = matr[e2r][e1c]
    return char1, char2
Cipher_text = 'RSSRDE'
Cipher_text = removeSpaces(toLowerCase(Cipher_text))
CipherList = Diagraph(FillerLetter(Cipher_text))
if len( CipherList[-1]) != 2:
   CipherList[-1] = CipherList[-1]+'z'
decryptedCipherList = decryptByPlayfairCipher(Matrix, CipherList)
DecipherText = ""
for i in decryptedCipherList:
    DecipherText += i
print("Deciphered Text:", DecipherText)
```

```
    Deciphered Text: attack
```



60009210033
Jhanvi Parekh
(DS) D-11

Date :
Page :

Plaintext : ATTACK
keyword : MONARCHY

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

ATTACK → AT TA ck

~~the~~ ciphertex → RS SR DE