

PRACTICAL-3

AIM:

The Rail Fence Cipher was invented in ancient times. It was used by the Greeks, who created a special tool, called scytale, to make message encryption and decryption easier. The letters are arranged in a way which is similar to the shape of the top edge of the rail fence. If king Leonidas want to send message to Sparta as “300 achieved glory at hot gate, unite for Greece” then what will be ciphertext when it is encrypted using 3 rows. Also implement decryption of message.

THEORY:

- The rail fence cipher is a form of transposition cipher.
- It derives its name from the way in which it is encoded.
- The algorithm is also known as ZIG-ZAG algorithm.

ENCRYPTION:

- In the rail fence cipher, the plain-text is written downwards and diagonally on successive rails of an imaginary fence.
- When we reach the bottom rail, we traverse upwards moving diagonally, after reaching the top rail, the direction is changed again.
- The alphabets of the message are written in a zig-zag manner.
- After each alphabet has been written, the individual rows are combined to obtain the cipher-text.

DECRYPTION:

- rail matrix can be constructed accordingly.
- Once we've got the matrix we can figure-out the spots where texts should be placed
- Then, we fill the cipher-text row wise. After filling it, we traverse the matrix in zig-zag manner to obtain the original text.

ADVANTAGES:

- There is a variable distance between consecutive letters
- The letters need not be arranged in fixed vertical columns that descends, but it can also be arranged in a zig zag manner.
- Therefore, this increases the difficulty of cracking the code.
- It is less prone to mistakes.

DISADVANTAGES:

- Once the attacker recognizes the method of encryption, it is very easy to crack the cipher text.
- As no extra characters are added and just original characters are re-arranged, it becomes easy to crack.
- The rail fence cipher is very easy to break as we only have to test all the possible divisors up to half the length of the text.

PROGRAM CODE:**LANGUAGE OF CODE:** PYTHON

```
#CS 346 -> CRNS
```

```
#PRACTICAL-3
```

```
#NAME: PARTH N PATEL
```

```
#ID: 19DCS098
```

```
#ALGORITHM NAME: RAIL FENCE CIPHER
```

```
#ENCRYPTION FUNCTION
```

```
#TWO PARAMETERS: PLAIN_TEXT AND KEY WHICH IS DEFAULT PARAMETER
```

```
#THE KEY=3 BECAUSE MENTIONED IN THE QUESTION TO TAKE ROW=3
```

```
#TO AVOID CONFUSION I TOOK KEY AS THIS ALGORITHM IS ALSO OF 2 TYPES:
```

```
#          1. KEY 2. KEYLESS
```

```
def encryption(plain_Text, key=3):
```

```
#MATRIX WHERE WE WILL STORE THE LETTERS OF THE MESSAGE
```

```
    rail_Matrix = [['\n' for i in range(len(plain_Text))]
```

```
                    for j in range(key)]
```

DIRECTION VARIABLE TO DETERMINE THE DIRECTION OF THE FLOW

direction = False

row=0

column = 0

#MAIN LOGIC

for i in range(len(plain_Text)):

 # FIRST, CHECK THE DIRECTION OF THE FLOW

 #IF WE HAVE FILLED THE TOP/BOTTOM ELEMENT OF THE COLUMN

 #THEN, CHANGE THE DIRECTION

 if (row == 0) or (row == key - 1):

 direction = not direction

 #FILL THE ALPHABET ON THE LOCATION

 rail_Matrix[row][column] = plain_Text[i]

 column += 1

 #WE WILL ALSO USE DIRECTION TO FIND THE NEXT ROW:

 #IF DIRECTION == TRUE, ROW IS INCREMENTED

 #ELSE ROW IS DECREMENTED.

 if direction:

 row += 1

```
    else:

        row -= 1

# ONCE THE MATRIX IS READY, WE CAN FIND THE CIPHER TEXT.

result = []

for i in range(key):

    for j in range(len(plain_Text)):

        if rail_Matrix[i][j] != '\n':

            result.append(rail_Matrix[i][j])

return("".join(result))


# CIPHER TEXT IS RETURNED IN THE FORM OF STRING


#DECRYPTION FUNCTION

def decryption(cipher_Text, key=3):


    #AGAIN, WE WILL CREATE THE CIPHER MATRIX FOR THE DECRYPTION
    PROCESS

    rail_Matrix = [['\n' for i in range(len(cipher_Text))]

                    for j in range(key)]


    direction = None

    row=0

    column = 0
```

```
# MARK THE PLACES WITH '#'
```

```
for i in range(len(cipher_Text)):
```

```
    if row == 0:
```

```
        direction = True
```

```
    if row == key - 1:
```

```
        direction = False
```

```
#PLACING THE MARKER
```

```
rail_Matrix[row][column] = '#'
```

```
column += 1
```

```
# FINDING THE NEXT ROW USING DIRECTION
```

```
if direction:
```

```
    row += 1
```

```
else:
```

```
    row -= 1
```

```
# FILL THE CIPHER MATRIX
```

```
index = 0
```

```
for i in range(key):
```

```
    for j in range(len(cipher_Text)):
```

```
        if ((rail_Matrix[i][j] == '#') and
```

```
            (index < len(cipher_Text))):
```

```
            rail_Matrix[i][j] = cipher_Text[index]
```

```
        index += 1

# THE READING THE MATRIX IN ZIG-ZAG MANNER TO GET THE PLAIN_TEXT
result = []

row, column = 0, 0

for i in range(len(cipher_Text)):

    # SAME DIRECTION RULES AS ENCRYPTION

    if row == 0:

        direction = True

    if row == key-1:

        direction = False

    # PLACING THE MARKER

    if (rail_Matrix[row][column] != '#'):

        result.append(rail_Matrix[row][column])

        column += 1

    # FINDING THE NEXT ROW

    if direction:

        row += 1

    else:

        row -= 1

return("".join(result))
```

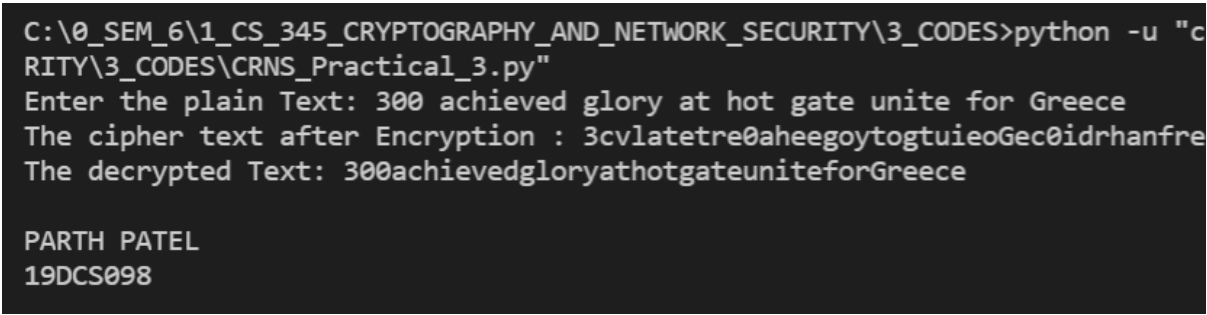
```
plain_Text=input("Enter the plain Text: ")
plain_Text=plain_Text.replace(" ", "")
cipher_Text=encryption(plain_Text)

print("The cipher text after Encryption : "+cipher_Text)

print("\nThe decrypted Text: "+decryption(cipher_Text))

print("\nPARTH PATEL\n19DCS098")
```

OUTPUT:



```
C:\0_SEM_6\1_CS_345_CRYPTOGRAPHY_AND_NETWORK_SECURITY\3_CODES>python -u "c
RITY\3_CODES\CRNS_Practical_3.py"
Enter the plain Text: 300 achieved glory at hot gate unite for Greece
The cipher text after Encryption : 3cvlatetre0aheegoytogtuieoGec0idrhanfre
The decrypted Text: 300achievedgloryathotgateuniteforGreece

PARTH PATEL
19DCS098
```

CONCLUSION:

- By performing the above practical, I learned about the rail fence cipher and how to decrypt it and its pros and cons.