

Exp: 1C**Rail Fence Cipher****Date: 10-02-2024****Aim:**

To write a python program implementing rail fence cipher algorithm

Algorithm:

1. Get the plain text from the user
2. Set the key as 2 by default.
3. Arrange the plaintext in two rows in a zig-zag manner.
4. Derive the cipher text by adding the first row of arrangement with the second row of arrangement.
5. Get the original text by using the cipher text and arranging it in zigzag manner and repeat the same process.

Program:

```
def encryptRailFence(text, key):
    rail = [['\n' for i in range(len(text))]for j in range(key)]
    dir_down = False
    row, col = 0, 0
    for i in range(len(text)):
        if (row == 0) or (row == key - 1):
            dir_down = not dir_down
        rail[row][col] = text[i]
        col += 1
        if dir_down:
            row += 1
        else:
            row -= 1
    result = []

    for i in range(key):
        for j in range(len(text)):
            if rail[i][j] != '\n':
                result.append(rail[i][j])
    return("".join(result))

def decryptRailFence(cipher, key):
    rail = [['\n' for i in range(len(cipher))]for j in range(key)]
    dir_down = None
```

```

row, col = 0, 0
for i in range(len(cipher)):
    if row == 0:
        dir_down = True
    if row == key - 1:
        dir_down = False
    rail[row][col] = '*'
    col += 1
    if dir_down:
        row += 1
    else:
        row -= 1
index = 0

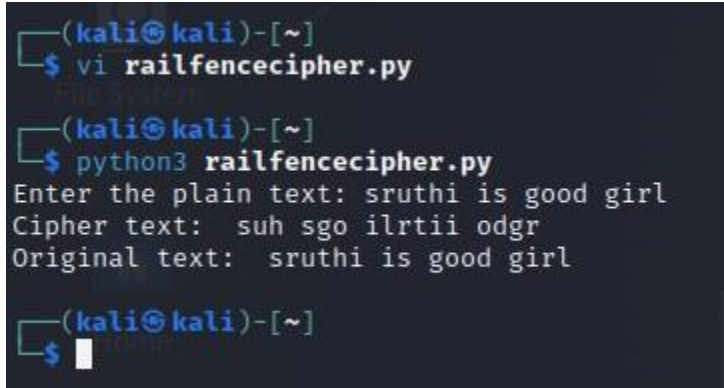
for i in range(key):
    for j in range(len(cipher)):
        if ((rail[i][j] == '*') and
            (index < len(cipher))):
            rail[i][j] = cipher[index]
            index += 1

result = []
row, col = 0, 0
for i in range(len(cipher)):
    if row == 0:
        dir_down = True
    if row == key-1:
        dir_down = False
    if (rail[row][col] != '*'):
        result.append(rail[row][col])
        col += 1
    if dir_down:
        row += 1
    else:
        row -= 1
    return("".join(result))

if __name__ == "__main__":
    pt=input("Enter plain text: ")
    ct=encryptRailFence(pt,2)

```

```
print(ct)
print(decryptRailFence(ct, 2))
```

Output:A terminal window with a dark background and light blue text. The prompt is '(kali㉿kali)-[~]'. The user enters '\$ vi railfencecipher.py'. The prompt changes to '(kali㉿kali)-[~]'. The user enters '\$ python3 railfencecipher.py'. The program outputs: 'Enter the plain text: sruthi is good girl', 'Cipher text: suh sgo ilrtii odgr', and 'Original text: sruthi is good girl'. The prompt returns to '(kali㉿kali)-[~]'. The user enters '\$' followed by a cursor.

```
(kali㉿kali)-[~]
$ vi railfencecipher.py

(kali㉿kali)-[~]
$ python3 railfencecipher.py
Enter the plain text: sruthi is good girl
Cipher text:  suh sgo ilrtii odgr
Original text:  sruthi is good girl

(kali㉿kali)-[~]
$
```

Result:

Thus the python program for rail fence cipher is implemented successfully.