Rail fence Encryption

```
def encrypt_rail_fence(text, num_rails):
    """
    Encrypt a message using the Rail Fence Cipher.

    Parameters:
    - text (str): The plaintext message to encrypt.
    - num_rails (int): The number of rails for the Rail Fence Cipher.

    Returns:
    - str: The encrypted message.
    """
    # Create the rail matrix initialized with newlines
    rail_matrix = [['\n' for _ in range(len(text))] for _ in range(num_rails)]

    # Set the initial direction and position
    down_direction = False
    row, col = 0, 0

    # Fill the rail matrix with the characters from the text
    for char in text:
        if row == 0 or row == num_rails - 1:
            down_direction = not down_direction

        rail_matrix[row][col] = char
        col += 1
        row += 1 if down_direction else -1

    # Construct the cipher text by reading the rail matrix row by row
    encrypted_text = []
    for r in range(num_rails):
        for c in range(len(text)):
            if rail_matrix[r][c] != '\n':
                encrypted_text.append(rail_matrix[r][c])

    return "".join(encrypted_text)

if __name__ == "__main__":
    text = input("Enter the text to encrypt: ")

    # Validate the number of rails input
    while True:
        try:
            num_rails = int(input("Enter the number of rails for encryption: "))
            if num_rails < 2:
                print("Number of rails must be at least 2. Please try again.")
                continue
            break
        except ValueError:
            print("Invalid input. Please enter a valid integer.")

    encrypted_text = encrypt_rail_fence(text, num_rails)
    print(f"Encrypted Message: {encrypted_text}")
```

```
PS I:\SEM 7\CSS\EXP\EXP-2> & C:/Users/Adnan/AppData/Local/Microsoft/WindowsApps/python3.11.exe "i:/SEM 7/CSS/EXP/EXP-2/encrypt_rail_fence.py"
Enter the text to encrypt: In a world increasingly driven by technology, the importance of cybersecurity cannot be overstated. As organizations expand the
ir digital footprints, they face growing threats from cyberattacks. Implementing robust security measures is crucial to protect sensitive data and maintai
n trust. From encryption to regular updates, proactive strategies are key to safeguarding information and ensuring operational resilience
Enter the number of rails for encryption: 6
Encrypted Message: I g li eneAaadotgr cmbieittaiFpeact f ouasndinlnboo meoscanvr sztpn iot h rhemcakeeourtrscaoeii nan rytrgdtataeekaegiinsrrtei lniyeynge
pcfrucoos. iixdrgfp,eeotaoytslnrsuyu ulrcsvadt .ori upeoirgresgnnt nieirlarcs v hyhon er t tdonoe ii rsycw trbt.pt tc sir ptnet nttmcoolusrvtiay uifaaenpo
 ie oraditc,tracbiy eaeran tetlit aigsfea mig emasct  e amirs nnta ,pese  oadomn gonlecwere tytbtgshanfn rInse osdaue r  strrd an
```

Rail fence Decryption

```
def decrypt_rail_fence(cipher, num_rails):
    """
    Decrypt a message encrypted with the Rail Fence
Cipher.

    Parameters:
    - cipher (str): The encrypted message to decrypt.
    - num_rails (int): The number of rails used during
encryption.

    Returns:
    - str: The decrypted plaintext message.
    """
    # Create a matrix to determine the pattern of rails
    rail_matrix = [['\n' for _ in range(len(cipher))] for
_ in range(num_rails)]

    # Set direction and position
    down_direction = None
    row, col = 0, 0

    # Mark the places with '*'
    for i in range(len(cipher)):
        if row == 0:
            down_direction = True
        elif row == num_rails - 1:
            down_direction = False

        rail_matrix[row][col] = '*'
        col += 1
        row += 1 if down_direction else -1

    # Fill the matrix with the cipher text
    index = 0
    for r in range(num_rails):
        for c in range(len(cipher)):
            if rail_matrix[r][c] == '*' and index <
len(cipher):
                rail_matrix[r][c] = cipher[index]
                index += 1
```

```
    # Read the matrix in a zigzag manner to decrypt the
message
    decrypted_text = []
    row, col = 0, 0
    for i in range(len(cipher)):
        if row == 0:
            down_direction = True
        elif row == num_rails - 1:
            down_direction = False

        if rail_matrix[row][col] != '*':
            decrypted_text.append(rail_matrix[row][col])
            col += 1

        row += 1 if down_direction else -1

    return "".join(decrypted_text)

if __name__ == "__main__":
    encrypted_text = input("Enter the encrypted text to
decrypt: ")
    key = int(input("Enter the number of rails for
decryption: "))
    decrypted_text = decrypt_rail_fence(encrypted_text,
key)
    print(f"Decrypted Message: {decrypted_text}")
```

```
PS I:\SEM 7\CSS\EXP\EXP-2> & C:/Users/Adnan/AppData/Local/Microsoft/WindowsApps/python3.11.exe "i:/SEM 7/CSS/EXP/EXP-2/decrypt_rail_fence.py"
Enter the encrypted text to decrypt: I g li eneAaadotgr cmbieittaiFpeact f ouasndinlnboo meoscanvr sztpn iot h rhemcakeeourtrscaoeii nan rytrgdtataee kaegiinsrrtei lniyeyngepcfrucoos. iixdrgfp,eeotaoytslnrsuyu ulrcsvadt .ori upeoirgresgnnt nieirlarcs v hyhon er t tdonoe ii rsycw trbt.pt tc sir ptne t nttmcoolusrvtiay uifaaenpo ie oraditc,tracbiy eaeran tetlit aigsfea mig emasct  e amirs nnta ,pese  oadomn gonlecwere tytbtgshanfn rInse osdaue r strrd an
Enter the number of rails for decryption: 6
Decrypted Message: In a world increasingly driven by technology, the importance of cybersecurity cannot be overstated. As organizations expand their digital footprints, they face growing threats from cyberattacks. Implementing robust security measures is crucial to protect sensitive data and maint ain trust. From encryption to regular updates, proactive strategies are key to safeguarding information and ensuring operational resilience
PS I:\SEM 7\CSS\EXP\EXP-2>
```

Eencrypt columnar Transposition

```python
import math

def encrypt_columnar_transposition(plaintext, key):
    key = str(key)
    msg_len = len(plaintext)
    num_cols = len(key)
    num_rows = math.ceil(msg_len / num_cols)

    padded_msg = plaintext + '_' * (num_cols * num_rows - msg_len)

    matrix = [padded_msg[i:i + num_cols] for i in range(0, len(padded_msg), num_cols)]

    sorted_key = sorted(key)
    cipher = []

    for sorted_col_idx in range(num_cols):
        actual_col_idx = key.index(sorted_key[sorted_col_idx])
        cipher.extend([matrix[row][actual_col_idx] for row in range(num_rows)])

    return "".join(cipher)

if __name__ == "__main__":
    plaintext = input("Enter the message to encrypt: ")
    key = input("Enter the key for encryption: ")

    if not key:
        print("Error: Key cannot be empty.")
    elif len(key) < 2:
        print("Error: Key must contain at least two characters.")
    else:
        encrypted_message = encrypt_columnar_transposition(plaintext, key)
        print(f"Encrypted Message: {encrypted_message}")
```

```
PS I:\SEM 7\CSS\EXP\EXP-2> & C:/Users/Adnan/AppData/Local/Microsoft/WindowsApps/python3.11.exe "i:/SEM 7/CSS/EXP/EXP-2/encrypt_columnar_transposition
.py"
Enter the message to encrypt: In a world increasingly driven by technology, the importance of cybersecurity cannot be overstated. As organizations ex
pand their digital footprints, they face growing threats from cyberattacks. Implementing robust security measures is crucial to protect sensitive dat
a and maintain trust. From encryption to regular updates, proactive strategies are key to safeguarding information and ensuring operational resilienc
e.
Enter the key for encryption: JaiHind
Encrypted Message: a sdbn pebrn asisd lrtcieorsegtissarsv itFcngp ia yfionn iecIrrleeg a e  r.gipeios rt yamtbcmsuocia it t reasi  aitdirlinleyncyinc
ccbs aoaitt,fohfbcpiuue c tttmn.eir scteksrni na eocgvtoetfsytedrtxhgotyg sctInoe ertesddasmpoato geou anreal_ da  h,mcyuaetAnnnrap awrreklnsraiip ia
a  noeu,trseadfoegtrn da  h,mcyuaetAnnnrap awrreklnsraiip iaa  noeu,trseadfoegtrnwnni lhrortoveoaetifne gt t ersyrc tn ntuoytlareertggmaupni.
```

Decrypt Columnar Transposition

```
import math

def decrypt_columnar_transposition(cipher, key):
    """
    Decrypt a message encrypted with the Columnar
Transposition Cipher.

    Parameters:
    - cipher (str): The encrypted message to decrypt.
    - key (str): The key used during the encryption.

    Returns:
    - str: The decrypted plaintext message.
    """
    msg_len = len(cipher)
    num_cols = len(key)
    num_rows = int(math.ceil(msg_len / num_cols))

    # Create a matrix to hold the decrypted text
    decrypted_matrix = [[''] * num_cols for _ in
range(num_rows)]

    # Determine the column order based on the sorted key
    sorted_key = sorted(key)

    index = 0
    for col in sorted_key:
        col_index = key.index(col)
        for row in range(num_rows):
            if index < msg_len:
                decrypted_matrix[row][col_index] =
cipher[index]
                index += 1

    # Flatten the matrix to retrieve the plaintext
    decrypted_message = ''.join(sum(decrypted_matrix,
[]))

    # Remove padding characters
    return decrypted_message.rstrip('_')
```

```
if __name__ == "__main__":
    encrypted_msg = input("Enter the encrypted message to
decrypt: ")
    key = input("Enter the key for decryption: ")
    decrypted_msg =
decrypt_columnar_transposition(encrypted_msg, key)
    print(f"Decrypted Message: {decrypted_msg}")
```

```
PS I:\SEM 7\CSS\EXP\EXP-2> & C:/Users/Adnan/AppData/Local/Microsoft/WindowsApps/python3.11.exe "i:/SEM 7/CSS/EXP/EXP-2/decrypt_columnar_transposition
.py"
Enter the encrypted message to decrypt: a sdbn pebrn asisd lrtcieorsegtissarsv itFcngp ia yfionn iecIrrleeg a e  r.gipeios rt yamtbcmsuocia it t reas
i  aitdirlinleyncyincccbs aoaitt,fohfbcpiuue c tttmn.eir scteksrni na eocgvtoetfsytedrtxhgotyg sctInoe ertesddasmpoato geou anreal_ da  h,mcyuaetAnnn
rap awrreklnsraiip iaa  noeu,trseadfoegtrn da  h,mcyuaetAnnnrap awrreklnsraiip iaa  noeu,trseadfoegtrnwnni lhrortoveoaetifne gt t ersyrc tn ntuoytlar
eertggmaupni.
Enter the key for decryption: JaiHind
Decrypted Message: In aworld ncreasngly diven b technlogy, he imprtanceof cybrsecurty canot be verstaed. Asorganiationsexpandtheir igitalfootprnts, t
ey fac growig threts fro cyberttacks Impleentingrobustsecuriy measres iscrucia to prtect snsitiv data nd maitain tust. Fom encyptionto reglar upates,
 roactie straegies re keyto safguardig infomationand enuring peratinal reilienc.
PS I:\SEM 7\CSS\EXP\EXP-2>
```