Single Columnar Cipher

Introduction

The Single Columnar Cipher is a classical encryption technique that organizes plaintext into a grid or table and rearranges the columns based on a key. It's a simple form of transposition cipher that provides basic data obfuscation.

How It Works

Key and Table Creation:

Choose a keyword or key phrase. The length of the key determines the number of columns in the grid.

Assign a numerical order to the letters of the key based on their alphabetical order (e.g., for the key KEY, the order is 2-1-3).

Encryption:

* Write the plaintext row by row into a grid/table.
* Rearrange the columns based on the numerical order of the key.
* Read the grid column by column in the new order to produce the ciphertext.

Decryption:

* Create a table with the ciphertext.
* Rearrange the columns back to their original order using the key.
* Read the plaintext row by row.

Example

Key: SECRET

Alphabetical order of SECRET: 3-1-4-2-5-6

Plaintext: MEETMEATNIGHT

Create the table:

S E C R E T M E E T M E A T N I G H T

Rearrange columns by the key order 3-1-4-2-5-6:

C S R E E T E M T E M E N A I T G H

Ciphertext (column by column): CEREMETNEIAHMT.

Python Implementation

Below is the Python code for encrypting and decrypting a Single Columnar Cipher:

def single\_columnar\_encrypt(plaintext, key):

# Remove spaces and convert to uppercase

plaintext = plaintext.replace(" ", "").upper()

key = key.upper()

# Generate the order of the columns based on the key

key\_order = sorted(list(key))

column\_order = [key.index(k) for k in key\_order]

# Determine number of rows

num\_columns = len(key)

num\_rows = len(plaintext) // num\_columns + (len(plaintext) % num\_columns != 0)

# Fill grid with plaintext

grid = [['' for \_ in range(num\_columns)] for \_ in range(num\_rows)]

idx = 0

for r in range(num\_rows):

for c in range(num\_columns):

if idx < len(plaintext):

grid[r][c] = plaintext[idx]

idx += 1

# Encrypt by reading columns in the order of the key

ciphertext = ''

for col in column\_order:

for row in grid:

if row[col]:

ciphertext += row[col]

return ciphertext

def single\_columnar\_decrypt(ciphertext, key):

key = key.upper()

num\_columns = len(key)

num\_rows = len(ciphertext) // num\_columns + (len(ciphertext) % num\_columns != 0)

# Generate the order of the columns based on the key

key\_order = sorted(list(key))

column\_order = [key.index(k) for k in key\_order]

# Create an empty grid to fill in the ciphertext

grid = [['' for \_ in range(num\_columns)] for \_ in range(num\_rows)]

# Fill the grid column by column using the column order

idx = 0

for col in column\_order:

for row in range(num\_rows):

if idx < len(ciphertext):

grid[row][col] = ciphertext[idx]

idx += 1

# Decrypt by reading row by row

plaintext = ''

for row in grid:

plaintext += ''.join(row)

return plaintext.strip()

# Example Usage

key = "SECRET"

plaintext = "MEET ME AT NIGHT"

ciphertext = single\_columnar\_encrypt(plaintext, key)

print("Ciphertext:", ciphertext)

decrypted\_text = single\_columnar\_decrypt(ciphertext, key)

print("Decrypted Text:", decrypted\_text)