1. Assume you intercepted the following ciphertext. Using a statistical attack, find the plaintext

"XLILSYWIMWRSAJSVWEPIJSVJSYVQMPPMSRHSPPEVWMXMWASVX-LQSVILYVVCFIJSVIXLIWIPPIVVIGIMZIWQSVISJJIVW"

from collections import Counter

# Ciphertext to analyze

ciphertext = "XLILSYWIMWRSAJSVWEPIJSVJSYVQMPPMSRHSPPEVWMXMWASVX-LQSVILYVVCFIJSVIXLIWIPPIVVIGIMZIWQSVISJJIVW"

# Frequency analysis function

def frequency\_analysis(text):

# Clean text: remove non-alphabetic characters and convert to upper case

text = ''.join(filter(str.isalpha, text)).upper()

# Count letter frequency

freq = Counter(text)

return freq.most\_common()

# Function to create a simple substitution based on frequency

def substitute\_text(ciphertext, substitution):

return ''.join(substitution.get(char, char) for char in ciphertext)

# Perform frequency analysis

freq = frequency\_analysis(ciphertext)

# Display the frequency of letters

print("Letter Frequency:")

for letter, count in freq:

print(f"{letter}: {count}")

# Hypothetical substitution mapping based on frequency analysis

substitution\_mapping = {

'X': 'T',

'L': 'H',

'I': 'E',

'S': 'A',

'Y': 'O',

'W': 'N',

'M': 'I',

'R': 'S',

'A': 'R',

'J': 'D',

'V': 'L',

'P': 'U',

'Q': 'Y',

'T': 'G', # Example (may not appear in the text)

# Add more mappings as needed

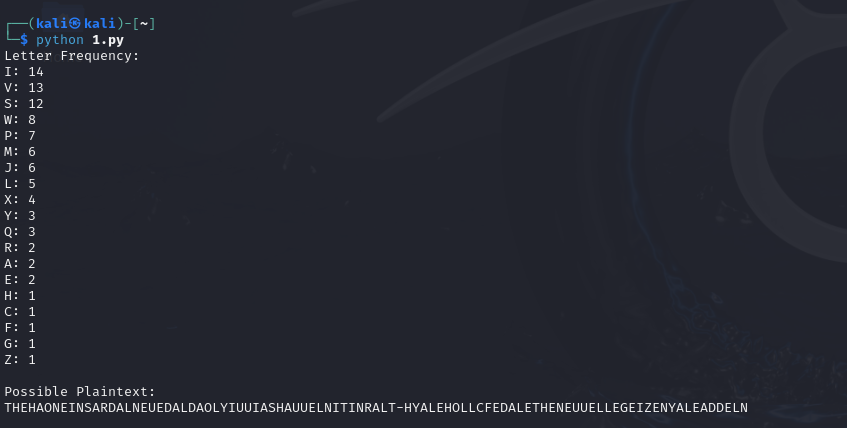
}

# Substitute the ciphertext using the mapping

plaintext = substitute\_text(ciphertext, substitution\_mapping)

print("\nPossible Plaintext:")

print(plaintext)



2. Write a Python script to encrypt using Rail Fence (Zig zag ) with three rows and with key (ONE).

def rail\_fence\_encrypt(plaintext, key):

# Number of rows for the Rail Fence Cipher

num\_rows = 3

# Create a list of strings for each row

rail = [''] \* num\_rows

row = 0

direction = 1 # 1 means moving down, -1 means moving up

# Fill the rail with the characters of the plaintext

for char in plaintext:

rail[row] += char

row += direction

# Change direction at the top and bottom rows

if row == num\_rows - 1 or row == 0:

direction \*= -1

# Combine the rows to get the ciphertext

ciphertext = ''.join(rail)

return ciphertext

key = "ONE"

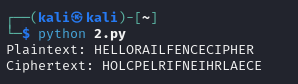
plaintext = "HELLORAILFENCECIPHER"

# Encrypt the plaintext using Rail Fence Cipher

ciphertext = rail\_fence\_encrypt(plaintext, key)

print("Plaintext:", plaintext)

print("Ciphertext:", ciphertext)



3. Write a python script to encrypt columnar transposition

def columnar\_transposition\_encrypt(plaintext, key):

# Remove spaces and convert to uppercase

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

# Determine the number of columns based on the key length

num\_cols = len(key)

# Calculate the number of rows needed

num\_rows = len(plaintext) // num\_cols + (len(plaintext) % num\_cols > 0)

# Create a grid to hold the characters

grid = [''] \* num\_cols

# Fill the grid with characters from plaintext

for i in range(len(plaintext)):

col = i % num\_cols

grid[col] += plaintext[i]

# Create a list of (key\_char, index) pairs

key\_indices = sorted(range(len(key)), key=lambda k: key[k])

# Generate the ciphertext by reading columns in the order of sorted key

ciphertext = ''

for index in key\_indices:

ciphertext += grid[index]

return ciphertext

key = "HACK"

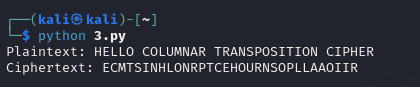
plaintext = "HELLO COLUMNAR TRANSPOSITION CIPHER"

# Encrypt the plaintext using Columnar Transposition

ciphertext = columnar\_transposition\_encrypt(plaintext, key)

print("Plaintext:", plaintext)

print("Ciphertext:", ciphertext)



4. Write a Python script to decrypt Rail Fence Cipher

def rail\_fence\_decrypt(ciphertext, num\_rows):

# Determine the length of the ciphertext

n = len(ciphertext)

# Create a grid to hold the characters

rail = [['\n' for \_ in range(n)] for \_ in range(num\_rows)]

# Fill the grid with placeholders based on the zigzag pattern

row, direction = 0, 1 # Start at the first row and move down

for i in range(n):

rail[row][i] = '\*'

row += direction

# Change direction at the top and bottom rows

if row == num\_rows - 1 or row == 0:

direction \*= -1

# Fill the grid with the ciphertext characters

index = 0

for r in range(num\_rows):

for i in range(n):

if rail[r][i] == '\*' and index < n:

rail[r][i] = ciphertext[index]

index += 1

# Read the plaintext from the grid in zigzag manner

result = []

row, direction = 0, 1 # Reset to the first row

for i in range(n):

result.append(rail[row][i])

row += direction

# Change direction at the top and bottom rows

if row == num\_rows - 1 or row == 0:

direction \*= -1

return ''.join(result)

ciphertext = "HOOLELCTENMRAIPIS"

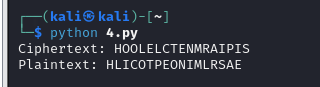
num\_rows = 3

# Decrypt the ciphertext using Rail Fence Cipher

plaintext = rail\_fence\_decrypt(ciphertext, num\_rows)

print("Ciphertext:", ciphertext)

print("Plaintext:", plaintext)



===============================================================================

"It does not matter how slowly you go so long as you do not stop." —Confucius

===============================================================================