1. Write a Python Program to perform brute force attack on the cipher text “dvvkzecfssprkkve"

def decrypt\_caesar\_cipher(ciphertext, shift):

decrypted\_text = []

for char in ciphertext:

if char.isalpha():

shift\_amount = shift % 26

if char.islower():

start = ord('a')

decrypted\_char = chr((ord(char) - start - shift\_amount) % 26 + start)

decrypted\_text.append(decrypted\_char)

else:

decrypted\_text.append(char) # Non-alphabetic characters are not changed

return ''.join(decrypted\_text)

def brute\_force\_caesar\_cipher(ciphertext):

print("Attempting to brute force Caesar cipher...")

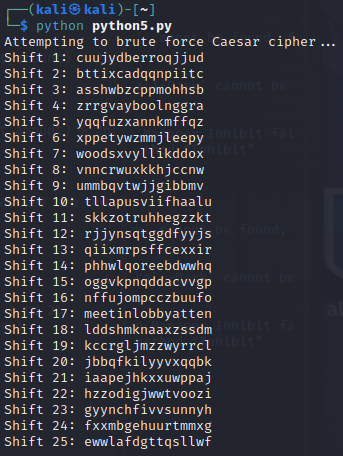
for shift in range(1, 26): # Try all possible shifts from 1 to 25

decrypted\_text = decrypt\_caesar\_cipher(ciphertext, shift)

print(f"Shift {shift}: {decrypted\_text}")

ciphertext = "dvvkzecfssprkkve"

brute\_force\_caesar\_cipher(ciphertext)



Plaintext : meet in lobby at ten

2. Write a Python program to use brute force attack to decipher the message.

Assume Affine cipher was used and "ab" is encrypted as "GL". Find the value of keys.

XPALASXYFGFUKPXUSOGEUTKCDGFXANMGNVS

3. Encrypt the plain text using Rail Fence cipher

rail\_fence\_encrypt(plaintext, num\_rails):

# Create a matrix with num\_rails rows

fence = [['' for \_ in range(len(plaintext))] for \_ in range(num\_rails)]

rail = 0

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

for char in plaintext:

fence[rail].append(char)

rail += direction

# Change direction at the top and bottom rails

if rail == 0 or rail == num\_rails - 1:

direction \*= -1

# Read off the rows to get the ciphertext

ciphertext = ''.join([''.join(row) for row in fence])

return ciphertext

# Example usage

plaintext = "HELLORAILFENCECIPHER"

num\_rails = 3

ciphertext = rail\_fence\_encrypt(plaintext, num\_rails)

print(f"Ciphertext: {ciphertext}")



4. Decrypt the cipher using Rail Fence

AAIUJ SIHBE KTEAO TEADE SNUTF EAEMR TAHSA

RHROA YHNFO AITTE EHCBO FVCAT RNMNS NUTFE

RASHL WFHLN HIUJS IHTKS OEHNI FISAE FNTIG

RMRSO LSTIS OKIEH PEOE

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

# Create a matrix with num\_rails rows

fence = [['' for \_ in range(len(ciphertext))] for \_ in range(num\_rails)]

# Determine the zigzag pattern

rail = 0

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

# Mark the positions with placeholder characters

for i in range(len(ciphertext)):

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

rail += direction

# Change direction at the top and bottom rails

if rail == 0 or rail == num\_rails - 1:

direction \*= -1

# Fill the fence with characters from ciphertext

index = 0

for r in range(num\_rails):

for c in range(len(ciphertext)):

if fence[r][c] == '\*':

fence[r][c] = ciphertext[index]

index += 1

# Read off the rails in a zigzag pattern to get the plaintext

plaintext = []

rail = 0

direction = 1

for i in range(len(ciphertext)):

plaintext.append(fence[rail][i])

rail += direction

# Change direction at the top and bottom rails

if rail == 0 or rail == num\_rails - 1:

direction \*= -1

return ''.join(plaintext)

# Example usage

ciphertext = ("AAIUJ SIHBE KTEAO TEADE SNUTF EAEMR TAHSA "

"RHROA YHNFO AITTE EHCBO FVCAT RNMNS NUTFE "

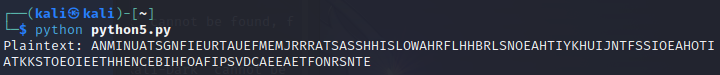
"RASHL WFHLN HIUJS IHTKS OEHNI FISAE FNTIG "

"RMRSO LSTIS OKIEH PEOE").replace(" ", "") # Removing spaces for simplicity

num\_rails = 4 # This should be the number of rails used during encryption

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

print(f"Plaintext: {plaintext}")



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

Never stop trying

Never stop believe

Never give up

Your day will come.

You only fail when you stop trying.

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