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...")

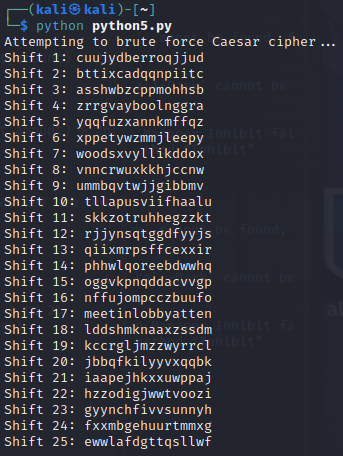
for shift in range(1, 26): # Try all possible shifts from 1 to 25

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

#!/usr/bin/python

import string

from sympy import mod\_inverse

def affine\_decrypt(ciphertext, a, b):

alphabet = string.ascii\_uppercase

m = len(alphabet)

a\_inv = mod\_inverse(a, m) # Find modular inverse of a modulo m

plaintext = ""

for char in ciphertext:

if char in alphabet:

y = alphabet.index(char)

x = (a\_inv \* (y - b)) % m

plaintext += alphabet[x]

else:

plaintext += char

return plaintext

def brute\_force\_affine(ciphertext, known\_plaintext, known\_ciphertext):

alphabet = string.ascii\_uppercase

m = len(alphabet)

known\_plaintext = known\_plaintext.upper()

known\_ciphertext = known\_ciphertext.upper()

# Create a dictionary of all possible (a, b) pairs

for a in range(1, m):

if gcd(a, m) != 1:

continue # 'a' must be coprime with m (i.e., 1 < a < 26)

for b in range(m):

# Encrypt known plaintext using (a, b) to check if it matches the known ciphertext

test\_ciphertext = ""

for char in known\_plaintext:

if char in alphabet:

x = alphabet.index(char)

y = (a \* x + b) % m

test\_ciphertext += alphabet[y]

else:

test\_ciphertext += char

if test\_ciphertext == known\_ciphertext:

# Found the correct (a, b) pair

print(f"Found keys: a = {a}, b = {b}")

# Decrypt the ciphertext using found keys

decrypted\_message = affine\_decrypt(ciphertext, a, b)

print(f"Decrypted message: {decrypted\_message}")

def gcd(a, b):

while b:

a, b = b, a % b

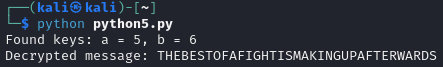
return a

ciphertext = "XPALASXYFGFUKPXUSOGEUTKCDGFXANMGNVS"

known\_plaintext = "ab"

known\_ciphertext = "GL"

brute\_force\_affine(ciphertext, known\_plaintext, known\_ciphertext)



3. Encrypt the plain text using Rail Fence cipher

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

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

rail = 0

direction = 1

for char in plaintext:

fence[rail].append(char)

rail += direction

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

direction \*= -1

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

return ciphertext

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):

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

rail = 0

direction = 1

for i in range(len(ciphertext)):

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

rail += direction

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

direction \*= -1

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

plaintext = []

rail = 0

direction = 1

for i in range(len(ciphertext)):

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

rail += direction

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

direction \*= -1

return ''.join(plaintext)

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

"RHROA YHNFO AITTE EHCBO FVCAT RNMNS NUTFE "

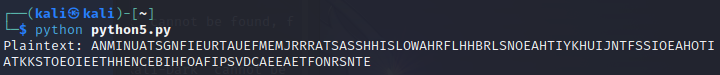
"RASHL WFHLN HIUJS IHTKS OEHNI FISAE FNTIG "

"RMRSO LSTIS OKIEH PEOE").replace(" ", "")

num\_rails = 4

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.

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