#Prodcut Cipher

import math

def format\_string(string):

return string.replace(" ", "").upper()

def cipherFunction(key, plain\_text):

cipher\_text = ""

for char in plain\_text:

if char.isupper():

cipher\_text += chr((ord(char)+key-65) % 26+65)

else:

cipher\_text += chr((ord(char)+key-97) % 26+97)

return cipher\_text

def encipher\_rail\_fence(text, depth):

length = len(text)

c = int(math.ceil(length/depth))

matrix = [[0 for i in range(c)] for i in range(depth)]

k = 0

ans = ""

for i in range(0, c):

for j in range(0, depth):

if k < length:

matrix[j][i] = text[k]

k = k+1

else:

matrix[j][i] = 'X'

for num in matrix:

for n in num:

ans += n

return ans

def decipher\_rail\_fence(text, depth):

length = len(text)

c = int(math.ceil(length/depth))

matrix = [[0 for i in range(c)] for i in range(depth)]

k = 0

ans = ""

for i in range(0, depth):

for j in range(0, c):

if k < length:

matrix[i][j] = text[k]

k = k+1

else:

matrix[i][j] = 'X'

for i in range(0, c):

for j in range(0, depth):

ans += matrix[j][i]

return ans

def printt(string):

for i in string:

print(i, end="")

print("\n1.Encrypt 2.Decrypt")

choice = int(input("Your choice: "))

if choice == 1:

print("\n1.Substitution->Transposition 2.Transposition->Substitution")

choice = int(input("Your choice: "))

if choice == 1:

plain\_text = str(input("Plain text: "))

key = int(input("Key: "))

temp = cipherFunction(key, plain\_text)

print("Caeser CT: {}".format(temp))

depth = int(input("Depth: "))

print("Cipher Text: {}".format(encipher\_rail\_fence(temp, depth)))

elif choice == 2:

plain\_text = str(input("Plain text: "))

depth = int(input("Depth: "))

temp = format\_string(plain\_text)

ct = encipher\_rail\_fence(temp, depth)

key = int(input("Key: "))

finalCt = cipherFunction(key, ct)

print("Cipher Text: {}".format(finalCt))

else:

print("Invalid input")

elif choice == 2:

print("\n1.Substitution->Transposition 2.Transposition->Substitution")

choice = int(input("Your choice: "))

if choice == 1:

plain\_text = str(input("Cipher text: "))

depth = int(input("Depth: "))

temp = decipher\_rail\_fence(plain\_text, depth)

key = int(input("Key: "))

finalPT = cipherFunction(-key, temp)

print("Plain Text: {}".format(finalPT))

elif choice == 2:

plain\_text = str(input("Cipher text: "))

depth = int(input("Depth: "))

key = int(input("Key: "))

temp = cipherFunction(-key, plain\_text)

finalPT = decipher\_rail\_fence(temp, depth)

print("Plain Text: {}".format(finalPT))

else:

print("Invalid input")

else:

print("Invalid input")

Output:

1.Encrypt 2.Decrypt

Your choice: 1

1.Substitution->Transposition 2.Transposition->Substitution

Your choice: 1

Plain text: mumbai

Key: 2

Caeser CT: owodck

Depth: 2

Cipher Text: oocwdk

1.Encrypt 2.Decrypt

Your choice: 2

1.Substitution->Transposition 2.Transposition->Substitution

Your choice: 1

Cipher text: oocwdk

Depth: 2

Key: 2

Plain Text: mumbai

1.Encrypt 2.Decrypt

Your choice: 1

1.Substitution->Transposition 2.Transposition->Substitution

Your choice: 2

Plain text: ciphertext

Depth: 3

Key: 3

Cipher Text: FKWWLHHASUAA

1.Encrypt 2.Decrypt

Your choice: 2

1.Substitution->Transposition 2.Transposition->Substitution

Your choice: 2

Cipher text: FKWWLHHASUAA

Depth: 3

Key: 3

Plain Text: CIPHERTEXTXX

#RSA

def gcd(n1,n2):

if n2 == 0:

return n1

return gcd(n2,n1%n2)

def RSA(data,mode):

p = 11

q = 13

n = p\*q

phi = (p-1)\*(q-1)

e = 2

while e<phi:

if gcd(e,phi)==1:

break

e+=1

d = pow(e,-1,phi)

if mode==0:

return pow(data,e)%n

elif mode==1:

return pow(data,d)%n

m = None

c = None

print("1.Encrypt 2.Decrypt")

choice = int(input("Choice: "))

if choice==1:

m = list(input("Message: "))

a = list(map(lambda char: ord(char.upper()),m))

c = [RSA(i,0) for i in a]

print("Cipher: {}".format(c))

elif choice==2:

c = input("Cipher: ").split(",")

temp = [RSA(int(i),1) for i in c]

m = "".join(list(map(lambda char: chr(char),temp)))

print("Message: {}".format(m))

Output:

Encryption

1.Encrypt 2.Decrypt

Choice: 1

Message: OUTPUT

Cipher: [40, 123, 72, 141, 123, 72]

Decryption

1.Encrypt 2.Decrypt

Choice: 2

Cipher: 40, 123, 72, 141, 123, 72

Message: OUTPUT

# Diffie Hellman

print("Public Keys")

P = int(input("User1 (P): "))

G = int(input("User2 (G): "))

print("-------------")

print("Private Keys")

a = int(input("User1 (a): "))

b = int(input("User2 (b): "))

x = pow(G,a)%P

y = pow(G,b)%P

Ka = pow(y,a)%P

Kb = pow(x,b)%P

print("-------------")

print("Shared key is {}".format(Ka))

Output:

Public Keys

User1 (P): 23

User2 (G): 9

-------------

Private Keys

User1 (a): 4

User2 (b): 3

-------------

Shared key is 9