Assignment No 5

Name :Prerana Gajare      Class:TEIT        RollNo:TI50

**TOPIC : RSA ALGORITHM FOR ENCRYPTION/DECRYPTION**

Source Code:

'''

WELCOME TO THE RSA ENCRYPTOR. THIS IS AN INTERACTIVE TOOL USED TO ENCRYPT OR DECRYPT A MESSAGE USING THE FAMOUS RSA ALGORITHM.

'''

import math

print("RSA ENCRYPTOR/DECRYPTOR")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

#Input Prime Numbers

print("PLEASE ENTER THE 'p' AND 'q' VALUES BELOW:")

p = int(input("Enter a prime number for p: "))

q = int(input("Enter a prime number for q: "))

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

#Check if Input's are Prime

'''THIS FUNCTION AND THE CODE IMMEDIATELY BELOW THE FUNCTION CHECKS WHETHER THE INPUTS ARE PRIME OR NOT.'''

def prime\_check(a):

    if(a==2):

        return True

    elif((a<2) or ((a%2)==0)):

        return False

    elif(a>2):

        for i in range(2,a):

            if not(a%i):

                return false

    return True

check\_p = prime\_check(p)

check\_q = prime\_check(q)

while(((check\_p==False)or(check\_q==False))):

    p = int(input("Enter a prime number for p: "))

    q = int(input("Enter a prime number for q: "))

    check\_p = prime\_check(p)

    check\_q = prime\_check(q)

#RSA Modulus

'''CALCULATION OF RSA MODULUS 'n'.'''

n = p \* q

print("RSA Modulus(n) is:",n)

#Eulers Toitent

'''CALCULATION OF EULERS TOITENT 'r'.'''

r= (p-1)\*(q-1)

print("Eulers Toitent(r) is:",r)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

#GCD

'''CALCULATION OF GCD FOR 'e' CALCULATION.'''

def egcd(e,r):

    while(r!=0):

        e,r=r,e%r

    return e

#Euclid's Algorithm

def eugcd(e,r):

    for i in range(1,r):

        while(e!=0):

            a,b=r//e,r%e

            if(b!=0):

                print("%d = %d\*(%d) + %d"%(r,a,e,b))

            r=e

            e=b

#Extended Euclidean Algorithm

def eea(a,b):

    if(a%b==0):

        return(b,0,1)

    else:

        gcd,s,t = eea(b,a%b)

        s = s-((a//b) \* t)

        print("%d = %d\*(%d) + (%d)\*(%d)"%(gcd,a,t,s,b))

        return(gcd,t,s)

#Multiplicative Inverse

def mult\_inv(e,r):

    gcd,s,\_=eea(e,r)

    if(gcd!=1):

        return None

    else:

        if(s<0):

            print("s=%d. Since %d is less than 0, s = s(modr), i.e., s=%d."%(s,s,s%r))

        elif(s>0):

            print("s=%d."%(s))

        return s%r

#e Value Calculation

'''FINDS THE HIGHEST POSSIBLE VALUE OF 'e' BETWEEN 1 and 1000 THAT MAKES (e,r) COPRIME.'''

for i in range(1,1000):

    if(egcd(i,r)==1):

        e=i

print("The value of e is:",e)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

#d, Private and Public Keys

'''CALCULATION OF 'd', PRIVATE KEY, AND PUBLIC KEY.'''

print("EUCLID'S ALGORITHM:")

eugcd(e,r)

print("END OF THE STEPS USED TO ACHIEVE EUCLID'S ALGORITHM.")

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("EUCLID'S EXTENDED ALGORITHM:")

d = mult\_inv(e,r)

print("END OF THE STEPS USED TO ACHIEVE THE VALUE OF 'd'.")

print("The value of d is:",d)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

public = (e,n)

private = (d,n)

print("Private Key is:",private)

print("Public Key is:",public)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

#Encryption

'''ENCRYPTION ALGORITHM.'''

def encrypt(pub\_key,n\_text):

    e,n=pub\_key

    x=[]

    m=0

    for i in n\_text:

        if(i.isupper()):

            m = ord(i)-65

            c=(m\*\*e)%n

            x.append(c)

        elif(i.islower()):

            m= ord(i)-97

            c=(m\*\*e)%n

            x.append(c)

        elif(i.isspace()):

            spc=400

            x.append(400)

    return x

#Decryption

'''DECRYPTION ALGORITHM'''

def decrypt(priv\_key,c\_text):

    d,n=priv\_key

    txt=c\_text.split(',')

    x=''

    m=0

    for i in txt:

        if(i=='400'):

            x+=' '

        else:

            m=(int(i)\*\*d)%n

#Message

message=input("What would you like encrypted or decrypted ?(seperate numbers with ',' for decryption) : ")

print("Your message is:",message)

#choose encrypt or decrypt to print

choose = input("Type '1' for encryption and '2' for decryption.")

if(choose=='1'):

  enc\_msg=encrypt(public,message)

  print("Your encrypted message is:",enc\_msg)

  print("Thank you for choosing RSA encryptor")

elif(choose=='2'):11

  print("Your decrypted message is:",decrypt(private,message))

  print("Thank you for choosing RSA decryptor")

else:

  print("You entered wrong option")

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*OUTPUT\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

RSA ENCRYPTOR/DECRYPTOR

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

PLEASE ENTER THE 'p' AND 'q' VALUES BELOW:

Enter a prime number for p: 12

Enter a prime number for q: 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Enter a prime number for p: 23

Enter a prime number for q: 3

RSA Modulus(n) is: 69

Eulers Toitent(r) is: 44

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The value of e is: 999

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EUCLID'S ALGORITHM:

44 = 0\*(999) + 44

999 = 22\*(44) + 31

44 = 1\*(31) + 13

31 = 2\*(13) + 5

13 = 2\*(5) + 3

5 = 1\*(3) + 2

3 = 1\*(2) + 1

END OF THE STEPS USED TO ACHIEVE EUCLID'S ALGORITHM.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

EUCLID'S EXTENDED ALGORITHM:

1 = 3\*(1) + (-1)\*(2)

1 = 5\*(-1) + (2)\*(3)

1 = 13\*(2) + (-5)\*(5)

1 = 31\*(-5) + (12)\*(13)

1 = 44\*(12) + (-17)\*(31)

1 = 999\*(-17) + (386)\*(44)

s=-17. Since -17 is less than 0, s = s(modr), i.e., s=27.

END OF THE STEPS USED TO ACHIEVE THE VALUE OF 'd'.

The value of d is: 27

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Private Key is: (27, 69)

Public Key is: (999, 69)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

What would you like encrypted or decrypted ?(seperate numbers with ',' for decryption) : Hello

Your message is: Hello

Type '1' for encryption and '2' for decryption.1

Your encrypted message is: [61, 13, 65, 65, 44]

Thank you for choosing RSA encryptor