**PRACTICAL-4**

**AIM:**

RSA algorithm is used by Salim to transfer session key to Anarkali. He suspects that Akbar is performing man in middle attack he chose to use 1024 bit prime numbers. Hint: you may choose to use big integer in java

**THEORY:**

* The **RSA algorithm** is an asymmetric cryptography algorithm; this means that it uses a public key and a private key (i.e two different, mathematically linked keys).
* As their names suggest, a public key is shared publicly, while a private key is secret and must not be shared with anyone.
* The RSA algorithm is named after those who invented it in 1978: Ron Rivest, Adi Shamir, and Leonard Adleman.
* It is safe for exchange of data over internet.

**ALGORITHM:**

* RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers
* The integers used by this method are sufficiently large making it difficult to solve.
* There are two sets of keys in this algorithm: private key and public key

**STEPS:**

1. Generate RSA Modulus
2. Find derived number (e)
3. Derive Private key and Public Key
4. Encryption

**NOTE:** We will also use Euler Totient, Multiplicative inverse and some other algorithms to make the calculation.

**ADVANTAGES:**

* RSA algorithm is safe and secure for transmitting confidential data.
* Cracking RSA algorithm is very difficult as it involves complex mathematics.
* Sharing public key to users is easy.

**DISADVANTAGES:**

* It has slow data transfer rate due to large numbers involved.
* It requires third party to verify the reliability of public keys sometimes.
* It requires some complex calculations both for encryption and decryption, which sometimes takes time and can causes delay and on the receiver’s side, calculations need to be done again.

**PROGRAM CODE:**

**LANGUAGE OF CODE:** Python

# CS 345 - CRNS

#PRACTICAL-4

#PERFORMED BY : PARTH N PATEL

# ID : 19DCS098

#LANGUAGE OF CODE: PYTHON

#IMPORTING math LIBRARY

import math

#TAKING INPUT FOR THE PRIME NUMBERS

print("PLEASE SELECT TWO PRIME NUMBERS: ")

p = int(input("ENTER THE FIRST PRIME NUMBER : "))

print()

q = int(input("ENTER THE SECOND PRIME NUMBER : "))

print()

#VALIDATING WHETHER THE NUMBERS ARE PRIME OR NOT

#FUNCTION TO VALIDATE IF NUMBER IS PRIME OR NOT

def validate\_Prime(x):

if(x==2):

#2 IS THE ONLY EVEN NUMBER

return True

elif((x<2) or ((x%2)==0)):

#NUMBERS LESS THAN 2 AND DIVISBLE BY ANOTHER NUMBER

# ARE NOT PRIME NUMBERS

return False

elif(x>2):

for i in range(2,x):

if not(x%i):

return False

return True

#CECKING IF P AND Q ARE PRIME OR NOT

check\_p = validate\_Prime(p)

check\_q = validate\_Prime(q)

#LOOP TILL PRIME NUMBERS ARE ENTERED

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 = validate\_Prime(p)

check\_q = validate\_Prime(q)

#MAIN LOGIC BUILDING FOR THE RSA

#FINDING THE RSA MODULUS

n = p \* q

print("RSA MODULUS : ",n)

print()

#EULER'S TOITENT

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

print("EULER'S TOITENT : ",r)

print()

#CALCULATING THE GCD

def find\_GCD(e,r):

while(r!=0):

e,r=r,e%r

return e

#EUCLID'S ALGORITHM

def euclid\_Algorithm(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 EUCLID'S ALGORITHM

def extended\_Euclid\_Algorithm(a,b):

if(a%b==0):

return(b,0,1)

else:

gcd,s,t = extended\_Euclid\_Algorithm(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 multiplicative\_Inverse(e,r):

gcd,s,\_=extended\_Euclid\_Algorithm(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

#CALCULATING THE VALUE OF e

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

for i in range(1,1000):

if(find\_GCD(i,r)==1):

e=i

print("THE VALUE OF e : ",e)

print()

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

print("EUCLID'S ALGORITHM:")

euclid\_Algorithm(e,r)

print()

print("EUCLID'S EXTENDED ALGORITHM:")

d = multiplicative\_Inverse(e,r)

print()

print("THE VALUE OF D : ",d)

print()

#PUBLIC KEY

public = (e,n)

#PRIVATE KEY

private = (d,n)

print("PRIVATE KEY : ",private)

print("PUBLIC KEY : ",public)

print()

#ENCRYPTION

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

#Message

plain\_Text = input("PLEASE ENTER THE MESSAGE TO BE ENCRYPTED : ")

print()

print("YOUR MESSAGE :",plain\_Text)

cipher\_Text=encrypt(public,plain\_Text)

print()

print("THE ENCRYPTED CIPHER TEXT : ")

print()

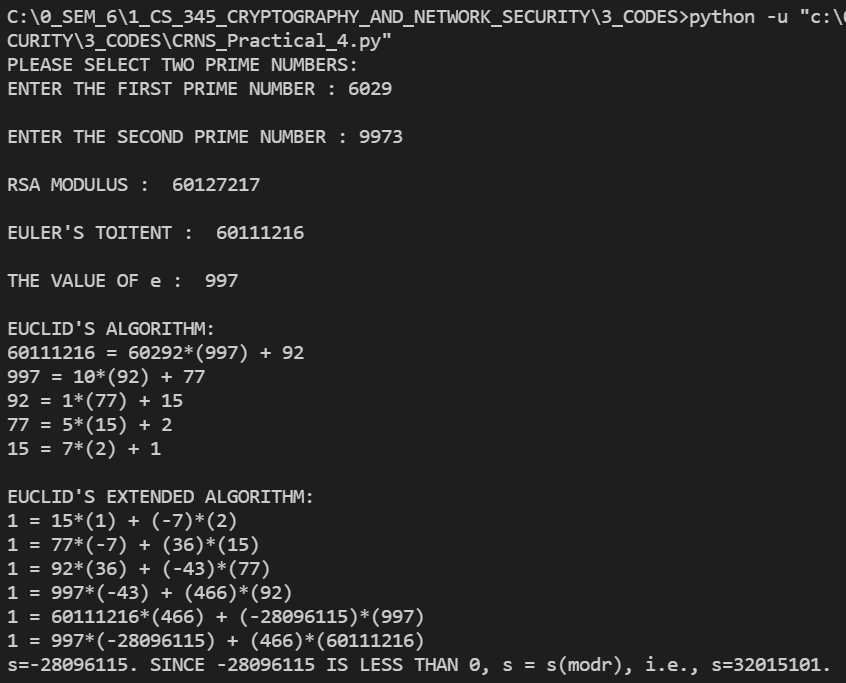
print(cipher\_Text)

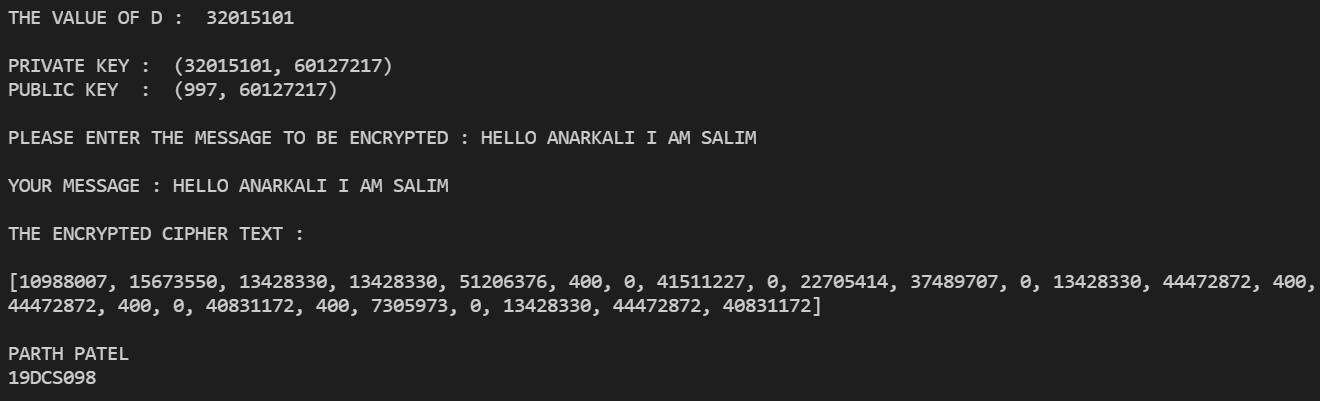
print()

print("PARTH PATEL\n19DCS098")

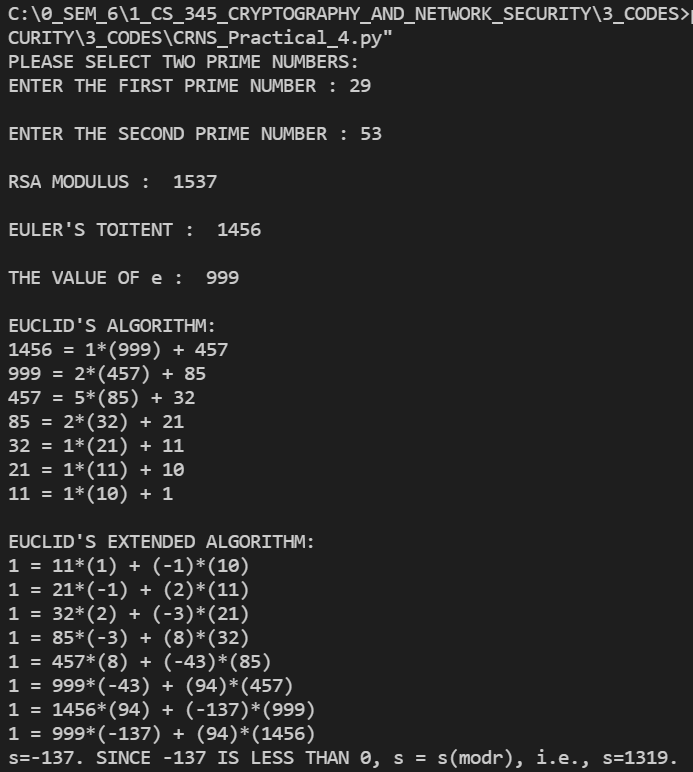
**OUTPUT:**

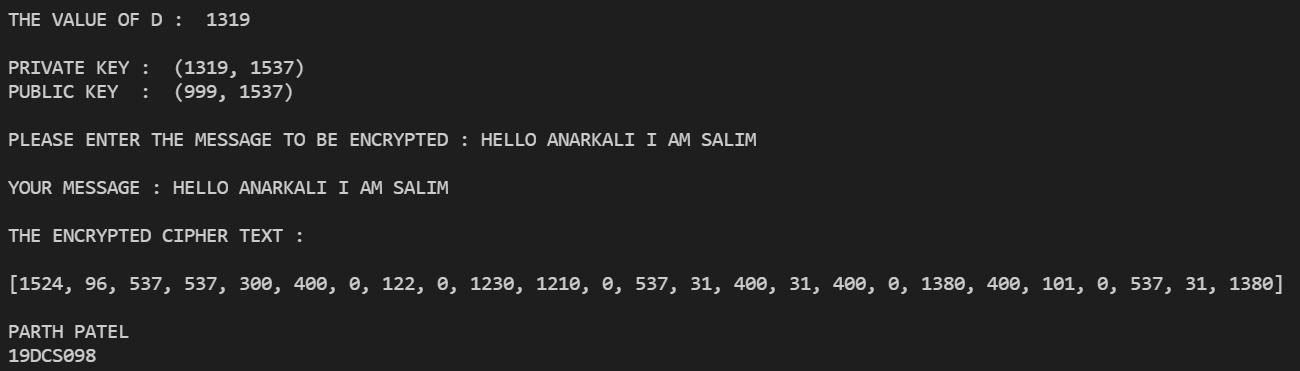
OUTPUT FOR BIG PRIME NUMBERS:





OUTPUT FOR SMALL PRIME NUMBERS:





**CONCLUSION:**

By performing the above practical, I learned the basic concept of RSA algorithm, why it is extensively used and how the encryption process works.