Name: Gaurav Kumar Singh

Registration Number: 19BCE2119

Course: Information Security Analysis and Audit (ISAA)

RSA Encryption

INTRODUCTION ABOUT ASYMMETRIC ENCRYPTION ALGORITHM

In Asymmetric cryptography algorithms the encryption key and decryption key are different, unlike Symmetric cryptography algorithms, where a single key is used to both encrypt and decrypt the messages. The two keys involved in asymmetric encryption algorithms are usually referred to as, public-key and private-key. Suppose a user ‘A’ wants to send message to another user ‘B’; both the users are assigned their own different sets of public and private keys. Public keys, as the name suggests are made public and everyone can see the public key. Private key however is tied to the user concerned and should not be made public. Since ‘A’ wants to send the message to ‘B’, ‘A’ encrypts the message using the public key of ‘B’ and sends the encrypted message to ‘B’. When ‘B’ receives the encrypted message, ‘B’ can use its private key which only ‘B’ knows about and decrypts the message using its private key. Since, no-one other than ‘B’ knows the private key of ‘B’ ideally, unless, the ‘B’s system is compromised, no one can read or decrypt the message except ‘B’.

Sender→Message→ →Encrypted Message→ →Decrypted Message→Recipient

Public Key Private Key

Asymmetric cryptographic algorithms are usually also slower than symmetric ones and require more computational power but are more secure.

Examples of asymmetric encryption algorithms are RSA, Diffie-Hellman, ECC, DSA, SHA-256 etc.

PSEUDOCODE:

```
int p = RandomPrime(a), q=RandomPrime(a) //a is the bit size of the prime number to be generated.

int n = p*q;

int phi = (p-1)*(q-1);

int e = FindRandomCoprime(phi);

int d = ModuloInverse(e,phi);

public_key = {e,n};

private_key = {d,n};

message = input();

message_arr = [];

for chars in message:

        message_arr.append(ascii(chars));
```

```
encrypted_arr=[]
//ENCRYPTION
for m in message_array:
        encrypted_arr.append((m^e)%n);
//DECRYPTION
decrypted_arr=[]
for m in encrypted_arr:
        decrypted_arr.append((m^d)%n);
decrypted_message="";
for m in decrypted_arr:
        decrypted_message += AsciiToChar(m);
print (decrypted_message);
```

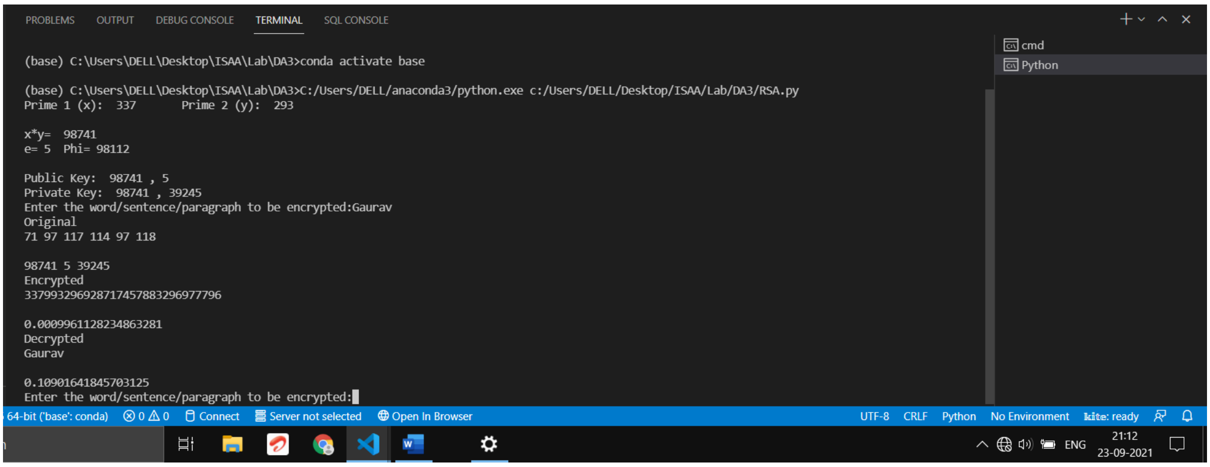
CODE (PYTHON):

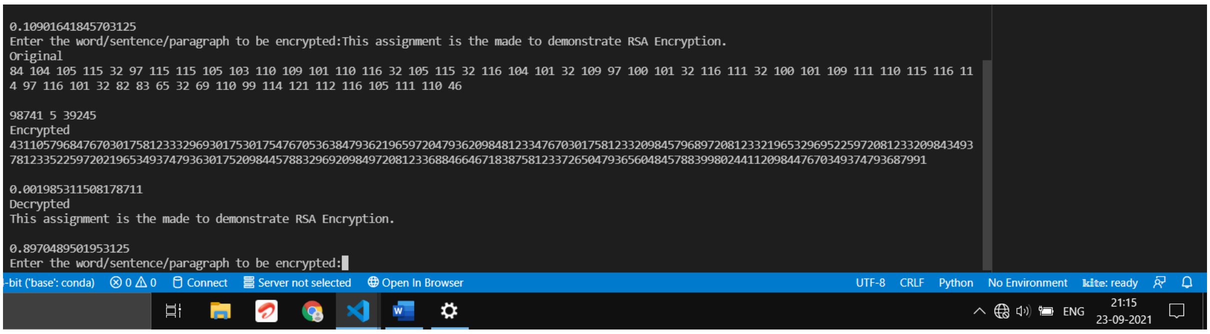
```
import numpy as np
import math
from Crypto.Util import number as crypt
import time
from tabulate import tabulate
# print(crypt.getPrime(3));
def egcd(a, b):
    if a == 0:
        return (b, 0, 1)
    g, y, x = egcd(b%a,a)
    return (g, x - (b//a) * y, y)
def modinv(a, m):
    g, x, y = egcd(a, m)
    if g != 1:
        raise Exception('No modular inverse')
    return x%m
x=crypt.getPrime(9);
y=crypt.getPrime(9);
```

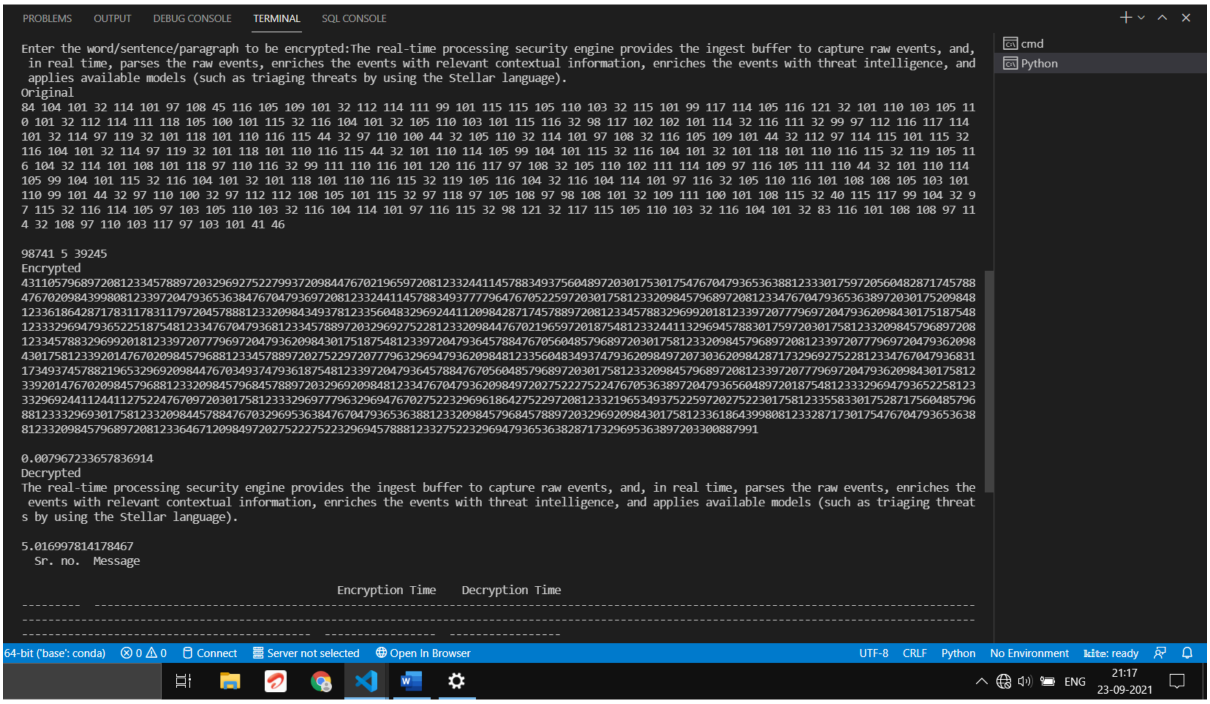
```
print("Prime 1 (x): ",x,"\tPrime 2 (y): ",y,"\n")


n=x*y
print("x*y= ",n)
totient =(x-1)*(y-1)
i=3
while(math.gcd(i,totient)!=1):
   i+=1
e=i
print("e=",e," Phi=",totient,"\n")
d=modinv(e,totient)
#print("d =",d)
print("Public Key: ",n,",",e)
print("Private Key: ",n,",",d)
public_key=[n,e]
# print((e*d)%totient)
for q in range(3):
   P=str(input("Enter the word/sentence/paragraph to be encrypted:"))
   start_time = time.time()
   Original=[]
   print("Original")
   for character in P:
      character=ord(character)
      Original.append(character)
      print(character,end=" ")


   print("\n")
   #ENCRYPTION
   print(n,e,d)
   Encrypted=[]
```

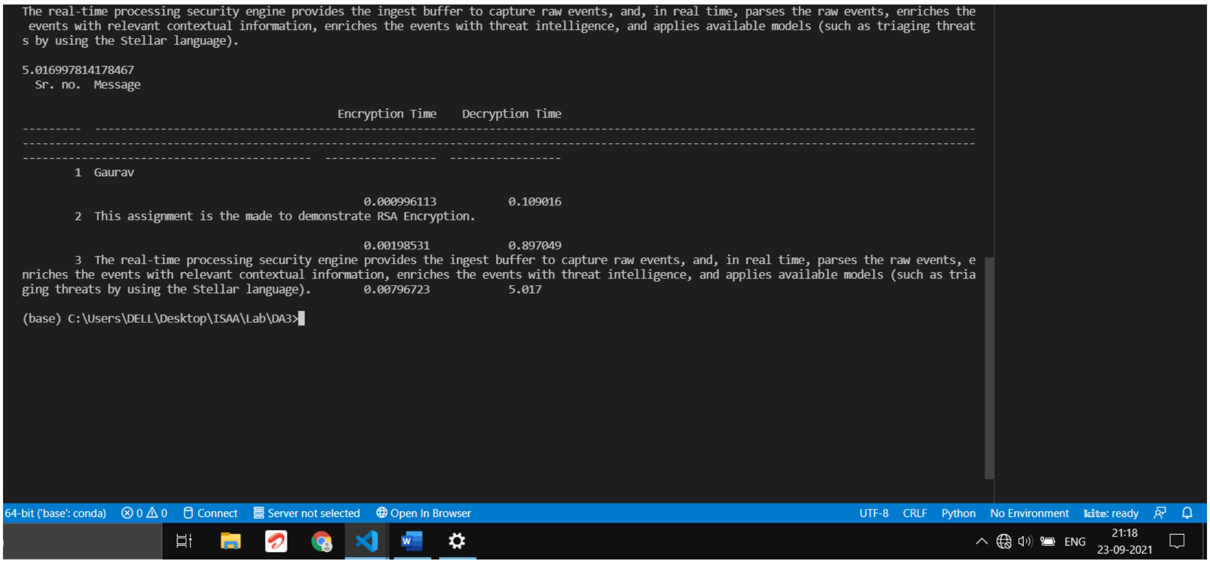
```
    print("Encrypted")
    for a in Original:
        temp=(a**e)%n
        Encrypted.append(temp)
        print(temp,end="")
    print("\n")
    encrypt_time=time.time()
    print(encrypt_time-start_time)
    #DECRYPTION
    Decrypted=[]
    i=0
    print("Decrypted")
    for a in Encrypted:
        temp=(a**d)%n
        Decrypted.append(temp)
        # print(temp,end=" ")
        i+=1
    for a in Decrypted:
        print(chr(a),end="")
    decrypt_time=time.time()
    print(decrypt_time-encrypt_time)
    if(q==0):
        words=[1,P,encrypt_time-start_time,decrypt_time-encrypt_time]
    if(q==1):
        sentence=[2,P,encrypt_time-start_time,decrypt_time-encrypt_time]
    if(q==2):
        paragraph=[3,P,encrypt_time-start_time,decrypt_time-encrypt_time]
print(tabulate([words,sentence,paragraph], headers=["Sr. no.","Message", "Encryption
Time","Decryption Time"]))
```

RESULTS:


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
(base) C:\Users\DELL\Desktop\ISAA\Lab\DA3>conda activate base
(base) C:\Users\DELL\Desktop\ISAA\Lab\DA3>C:/Users/DELL/anaconda3/python.exe c:/Users/DELL/Desktop/ISAA/Lab/DA3/RSA.py
Prime 1 (x): 337 Prime 2 (y): 293
x*y= 98741
e= 5 Phi= 98112
Public Key: 98741 , 5
Private Key: 98741 , 39245
Enter the word/sentence/paragraph to be encrypted:Gaurav
Original
71 97 117 114 97 118
98741 5 39245
Encrypted
33799329692871745788329697 7796
0.0009961128234863281
Decrypted
Gaurav
0.10901641845703125
Enter the word/sentence/paragraph to be encrypted:
cmd
Python
64-bit ('base': conda)
Connect
Server not selected
Open In Browser
UTF-8 CRLF Python No Environment kite: ready
ENG
21:12
23-09-2021


0.10901641845703125
Enter the word/sentence/paragraph to be encrypted:This assignment is the made to demonstrate RSA Encryption.
Original
84 104 105 115 32 97 115 115 105 103 110 109 101 110 116 32 105 115 32 116 104 101 32 109 97 100 101 32 116 111 32 100 101 109 111 110 115 116 11
4 97 116 101 32 82 83 65 32 69 110 99 114 121 112 116 105 111 110 46
98741 5 39245
Encrypted
0.001985311508178711
Decrypted
This assignment is the made to demonstrate RSA Encryption.
0.8970489501953125
Enter the word/sentence/paragraph to be encrypted:
-bit ('base': conda)
Connect
Server not selected
Open In Browser
UTF-8 CRLF Python No Environment kite: ready
ENG
21:15
23-09-2021


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
Enter the word/sentence/paragraph to be encrypted:The real-time processing security engine provides the ingest buffer to capture raw events, and, in real time, parses the raw events, enriches the events with relevant contextual information, enriches the events with threat intelligence, and applies available models (such as triaging threats by using the Stellar language).
Original
98741 5 39245
Encrypted
0.007967233657836914
Decrypted
The real-time processing security engine provides the ingest buffer to capture raw events, and, in real time, parses the raw events, enriches the events with relevant contextual information, enriches the events with threat intelligence, and applies available models (such as triaging threats by using the Stellar language).
5.016997814178467
Sr. no. Message
Encryption Time Decryption Time
cmd
Python
64-bit ('base': conda)
Connect
Server not selected
Open In Browser
UTF-8 CRLF Python No Environment kite: ready
ENG
21:17
23-09-2021



COMPARITIVE CHART:

Sr. No.	**Message**	**Encryption Time (In seconds)**	**Decryption Time (In second)**
1	Gaurav	0.000996113	0.109016
2	This assignment is the made to demonstrate RSA Encryption.	0.00198531	0.897049
3	The real-time processing security engine provides the ingest buffer to capture raw events, and, in real time, parses the raw events, enriches the events with relevant contextual information, enriches the events with threat intelligence, and applies available models (such as triaging threats by using the Stellar language).	0.00796723	5.017