

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'.



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

337993296928717457883296977796

0.0009961128234863281

Decrypted

Gaurav

0.10901641845703125

Enter the word/sentence/paragraph to be encrypted:

64-bit (base: conda)

0 0 0

Connect

Server not selected

Open In Browser

UTF-8

CRLF

Python

No Environment

Idle: ready

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

431105796847670301758123329693017530175476705363847936219659720479362098481233476703017581233209845796897208123321965329695225972081233209843493

781233522597202196534937479363017520984457883296920984972081233688466467183875812337265047936560484578839980244112098447670349374793687991

0.001985311508178711

Decrypted

This assignment is the made to demonstrate RSA Encryption.

0.8970489501953125

Enter the word/sentence/paragraph to be encrypted:

64-bit (base: conda)

0 0 0

Connect

Server not selected

Open In Browser

UTF-8

CRLF

Python

No Environment

Idle: ready

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

84 104 101 32 114 101 97 108 45 116 105 109 101 32 112 114 111 99 101 115 115 105 110 103 32 115 101 99 117 114 105 116 121 32 101 110 103 105 11

0 101 32 112 114 111 118 105 100 101 115 32 116 104 101 32 105 110 103 101 115 116 32 98 117 102 102 101 114 32 116 111 32 99 97 112 116 117 114

101 32 114 97 119 32 101 118 101 110 116 115 44 32 97 110 100 44 32 105 110 32 114 101 97 108 32 116 105 109 101 44 32 112 97 114 115 101 115 32

116 104 101 32 114 97 119 32 101 118 101 110 116 115 44 32 101 110 114 105 99 104 101 115 32 116 104 101 32 101 118 101 110 116 115 32 119 105 11

6 104 32 114 101 108 101 118 97 110 116 32 99 111 110 116 101 120 116 117 97 108 32 105 110 102 111 114 109 97 116 105 111 110 44 32 101 110 114

105 99 104 101 115 32 116 104 101 32 101 118 101 110 116 115 32 119 105 116 104 32 116 104 114 101 97 116 32 105 110 116 101 108 108 105 103 101

110 99 101 44 32 97 110 100 32 97 112 112 108 105 101 115 32 97 118 97 105 108 97 98 108 101 32 109 111 100 101 108 115 32 40 115 117 99 104 32 9

7 115 32 116 114 105 97 103 105 110 103 32 116 104 114 101 97 116 115 32 98 121 32 117 115 105 110 103 32 116 104 101 32 83 116 101 108 108 97 11

4 32 108 97 110 103 117 97 103 101 41 46

98741 5 39245

Encrypted

4311057968972081233457889720329692752279937209844767021965972081233244114578834937560489720301753017547670479365363881233301759720560482871745788

476702098439988812339720479365363847670479369720812332441145788349377796476705225972030175812332098457968972081233476704793653638972030175209848

123361864287178311783117972045788812332098434937812335604832969244112098428717457889720812334578832969920181233972077969720479362098430175187548

1233329694793652251875481233476704793681233457889720329692752281233209844767021965972018754812332441132969457883017597203017581233209845796897208

12334578832969920181233972077969720479362098430175187548123397204793645788476705604857968972030175812332098457968972081233972077969720479362098

4301758123320147670209845796881233457889720275297207796329604793620984812335604834937479362098497207303620984287173296927522812334767047936831

1720374578021965329692098447670301747936187548123307204793645788476705604857968972030175812332098457968972081233972077969720479362098430175812

33920147670209845796881233209845706845788972032969209848123347670479362098497207252275224767053638972047936560489720187548123329694793652258123

332969244112441127522476709720301758123332969779632969476702752232969618642752297208123321965349375225972027522301758123355833017528717560485796

8812333296930175812332098445788476703296953638476704793653638812332098457968457889720329692098430175812336186439988812332871730175476704793653638

812332098457968972081233646712098497202752275223296945788812332752232969479365363828717329695363897203300887991

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 threat s by using the Stellar Language).

5.016997814178467

Sr. no. Message

Encryption Time Decryption Time

64-bit (base: conda)

0 0 0

Connect

Server not selected

Open In Browser

UTF-8

CRLF

Python

No Environment

Idle: ready

21:17

23-09-2021

```

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 threat
s by using the Stellar language).

5.016997814178467
Sr. no. Message

Encryption Time    Decryption Time
-----
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, e
nriches the events with relevant contextual information, enriches the events with threat intelligence, and applies available models (such as tria
ging threats by using the Stellar language).
0.00796723         5.017

(base) C:\Users\DELL\Desktop\ISAA\Lab\DA3>

```

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