Lab – 4

Subject : NIS

Aim :- Implement RSA algorithm.

1. Do Key generation 2. Encryption 3. Decryption . Use only Multiply and Square to do Modular Exponentiation. One can use Miller Rabin for Primality Testing (Optional but recommended) . Use large prime numbers due to security concerns.

Description :-

- RSA means (Rivest-Shamir-Adleman) is an algorithm used by morden era to encrypt and decrypt the plain messages. It is an asymmetric cryptographic algorithm. It means there are two different keys. And in this algorithm the encryption take place by public key that's why this also called public key cryptography.
- The RSA algorithm ensures that the keys, in the above illustration, are as secure as possible. The following steps highlight how it works:

=>Generating the Key:

- Select two large prime numbers, p and q. The prime numbers need to be large so that they will be difficult for someone to figure out.
- Calculate the n = p * q.
- Calculate the totient function phi(n) = phi(p) * phi(q)
- Select an integer e, such that e is co-prime to phi(n) and 1 <= e < phi(n). the pair (e, n) makes up the public key.
- Calculate the d such that e * d = 1 mod phi(n).
- Here d can be found using the extended Euclidean algorithm. The pair (d, n) makes up the private key.

=>Encryption:

- Encryption done by the public key (e, n).
- Cipher text = P ^ e mod n.

=> Decryption:

- Decryption done by the private key (d, n).
- Plain text = c ^ d mod n.

Program: -

```
from random import randint

from ExtendedEuclidian import multiplicative_inverse as mi

from MultiplyAndSquare import multiply_and_square

def phi(a):

    return a-1

def encryption(m,e,n):

    return multiply_and_square(m,e,n)

def decryption(c,d,n):

    return multiply_and_square(c,d,n)

def generate_keys(p,q):

    e=randint(2,(phi(p)*phi(q))-1)

    gcd,_=mi(e,phi(p)*phi(q))

    while(gcd != 1):

        e=randint(2,(phi(p)*phi(q))-1)

        gcd,_=mi(e,phi(p)*phi(q))

    _,d=mi(e,phi(p)*phi(q))

    public=(e,p*q)

    private=(d,p*q)

    return public,private
```

```
if __name__ == "__main__":

    p,q=list(map(int,input("Enter p q : ").split()))

    if(p == q):

        q=input("p and q can't be same so again Enter q : ")

    public,private=generate_keys(p,q)

    encrypted_txt=encryption(int(input("Enter plain text :
")),public[0],public[1])

    print("Encrypted text : "+str(encrypted_txt))

    decrypted_txt=decryption(encrypted_txt,private[0],private[1])

    print("Decrypted text : "+str(decrypted_txt))
```

Output: -

```
PS D:\DDIT CE\Sem 6\NIS\Lab 4> python .\RSAAlgo.py
Enter p q : 29 37
Enter plain text : 123
Encrypted text : 194
Decrypted text : 123
PS D:\DDIT CE\Sem 6\NIS\Lab 4> python .\RSAAlgo.py
Enter p q : 31 67
Enter plain text : 23
Encrypted text : 1976
Decrypted text : 23
PS D:\DDIT CE\Sem 6\NIS\Lab 4>
```