CRYPTOGRAPHY AND NETWORK SECURITY
LAB - 5

Name of the Student: SreeDananjay S
Registration Number: 21BAI1807
Slot: L31+L32
Course Code: BCSE309P
Programme: Bachelor of Technology in Computer Science and Engineering with Specialization in Artificial Intelligence and Machine Learning
School: School of Computer Science and Engineering(SCOPE)

Q) Implement RSA algorithm for the following conditions

a) If P and Q are given

Code:

```
import random
import math

def gcd(a, b):
   while b != 0:
      a, b = b, a % b
   return a

def mod_inverse(e, phi):
   d_old, d_new = 0, 1
   r_old, r_new = phi, e

   while r_new != 0:
      quotient = r_old // r_new
      d_old, d_new = d_new, d_old - quotient * d_new
      r_old, r_new = r_new, r_old - quotient * r_new

   if d_old < 0:
      d_old += phi

   return d_old
```

```
def generate_factors_n(n):

  factors=list()
  while n % 2 == 0:
    factors.append(2)
    n = n / 2

  for i in range(3,int(math.sqrt(n))+1,2):

    while n % i== 0:
      factors.append(i)
      n = n / i

  if n > 2:
    factors.append(n)

  return factors

def generate_keys(pq_given=True):
  if pq_given:
    p = 61
    q = 53

    n = p * q

    phi = (p - 1) * (q - 1)
```

```
  else:
    n=187
    factors=generate_factors_n(n)
    p=int(factors[0])
    q=int(factors[1])

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

  e = 3
  while gcd(e, phi) != 1:
    e += 2

  d = mod_inverse(e, phi)

  return ((e, n), (d, n))

def encrypt_message(public_key, message):
  e, n = public_key
  encrypted_message = [pow(ord(char), e, n) for char in message]
  return encrypted_message

def decrypt_message(private_key, encrypted_message):
  d, n = private_key
  decrypted_message = ''.join([chr(pow(char, d, n)) for char in encrypted_message])
  return decrypted_message
```

```
if __name__ == "__main__":
   public_key, private_key = generate_keys(pq_given=True)
   print("Public Key:", public_key)
   print("Private Key:", private_key)

   message = input("Enter message: ")

   encrypted_message = encrypt_message(public_key, message)
   print("Encrypted Message:", encrypted_message)

   decrypted_message = decrypt_message(private_key, encrypted_message)
```

Output Screenshots:

```
➜  Lab5-RSA python3 RSA.py
Public Key: (7, 3233)
Private Key: (1783, 3233)
Enter message: SreeDananjay
Encrypted Message: [1825, 1797, 3071, 3071, 2056, 1818, 1544, 1818, 1544, 2120, 1818, 731]
Decrypted Message: SreeDananjay
➜  Lab5-RSA
```

b) If N value is given

Code:

```
import random
import math


def gcd(a, b):
   while b != 0:
      a, b = b, a % b
   return a
```

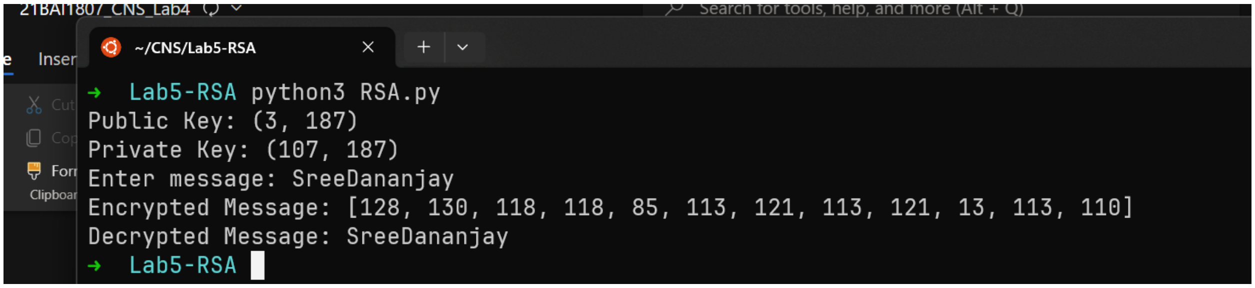
```
def mod_inverse(e, phi):
    d_old, d_new = 0, 1
    r_old, r_new = phi, e

    while r_new != 0:
        quotient = r_old // r_new
        d_old, d_new = d_new, d_old - quotient * d_new
        r_old, r_new = r_new, r_old - quotient * r_new

    if d_old < 0:
        d_old += phi

    return d_old

def generate_factors_n(n):

    factors=list()
    while n % 2 == 0:
        factors.append(2)
        n = n / 2

    for i in range(3,int(math.sqrt(n))+1,2):

        while n % i== 0:
            factors.append(i)
```

```
        n = n / i

    if n > 2:
        factors.append(n)

    return factors

def generate_keys(pq_given=True):
    if pq_given:
        p = 61
        q = 53

        n = p * q

        phi = (p - 1) * (q - 1)
    else:
        n=187
        factors=generate_factors_n(n)
        p=int(factors[0])
        q=int(factors[1])

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

    e = 3
    while gcd(e, phi) != 1:
        e += 2
```

```
    d = mod_inverse(e, phi)

    return ((e, n), (d, n))

def encrypt_message(public_key, message):
    e, n = public_key
    encrypted_message = [pow(ord(char), e, n) for char in message]
    return encrypted_message

def decrypt_message(private_key, encrypted_message):
    d, n = private_key
    decrypted_message = ''.join([chr(pow(char, d, n)) for char in encrypted_message])
    return decrypted_message

if __name__ == "__main__":
    public_key, private_key = generate_keys(pq_given=False)
    print("Public Key:", public_key)
    print("Private Key:", private_key)

    message = input("Enter message: ")

    encrypted_message = encrypt_message(public_key, message)
    print("Encrypted Message:", encrypted_message)

    decrypted_message = decrypt_message(private_key, encrypted_message)
```

```
print("Decrypted Message:", decrypted_message)
```

Output Screenshots:



Result:

Thus, the RSA algorithm has been successfully executed and verified under both conditions.