CS1702 Network Security : Assignment
RSA Algorithm

- Tharaneeshwaran V U (CS22B1056)

1. Introduction

This project demonstrates the implementation of the RSA encryption and decryption algorithm. RSA is a widely-used asymmetric cryptographic technique that uses two keys: a public key for encryption and a private key for decryption. The algorithm ensures secure data transmission and underpins the security of modern communication protocols like HTTPS, TLS, and PGP.

RSA relies on the mathematical difficulty of factoring large prime numbers. The public key is shared openly, while the private key remains confidential. Messages encrypted with the public key can only be decrypted with the private key, ensuring confidentiality. Conversely, encrypting with the private key enables digital signature verification.

2. Code

Helper functions for string/decimal conversion

```
def str2dec(st):
    return [ord(i) for i in st]

def dec2str(dec):
    return ''.join(chr(i) for i in dec)
```

Prime number generator

```
def generate_prime(beg=1000, end=10000):
    import random
    import math
    beg_rand = random.randint(beg, end)
    if beg_rand % 2 == 0:
        beg_rand += 1
    for possiblePrime in range(beg_rand, end, 2):
        isPrime = True
        for num in range(3, math.floor(possiblePrime/2), 2):
            if possiblePrime % num == 0:
                isPrime = False
        if isPrime:
            return possiblePrime
```

Key generation functions

```
def generate_nkey(p, q):
    return p * q

def generate_ekey(p, q):
    phi = (p-1)*(q-1)
    for e in range(3, phi, 2):
        if math.gcd(e, phi) == 1:
            return e

def generate_dkey(e):
    phi = (p-1)*(q-1)
    d = int(phi / e)
    while (d * e) % phi != 1:
        d += 1
    return d
```

Encryption and Decryption function

```
def endecrypt_message(m, key, n):
    res = 1
    m = m % n
    if m == 0:
        return 0
    while key > 0:
        if key & 1:
            res = (res * m) % n
        key = key >> 1
        m = (m * m) % n
    return res
```

3. Output

Two random prime numbers:
Prime 1: <p>
Prime 2: <q>

Public key {e, n} = {<e>, <n>}
Private key {d, n} = {<d>, <n>}

Original Message: 'Hola, This is Tharan!'
Decimal Representation: [H, o, l, a, ...]

Encrypted (using public key): [<cipher>...]
Decrypted (using private key): 'Hola, This is Tharan!'
Encrypted (using private key): [<signature>...]
Decrypted (using public key): 'Hola, This is Tharan!'

```
Original String:         Hola
String into Decimal:     [72, 111, 108, 97]
Decimal into String:     Hola
Two random prime numbers
        Prime 1:  4831
        Prime 2:  2969

n key:  14343239
e key:  3438809
d key:  5233289

Public key {e, n}: {3438809, 14343239}
Private key {d, n}: {5233289, 14343239}

MESSAGE
         Hola, This is Tharan!
         [72, 111, 108, 97, 44, 32, 84, 104, 105, 115, 32, 105, 115, 32, 84, 104, 97, 114, 97, 110, 33]


ENCRYPTED
         [819578, 9458260, 6765241, 5698664, 1058151, 12629752, 11829733, 7367047, 13902965, 11496679, 12629752, 13902965, 11496679, 12629752, 11829733, 7367047, 5698664, 2795056, 5698664, 314208, 11635956]


DECRYPTED
         Hola, This is Tharan!
         [72, 111, 108, 97, 44, 32, 84, 104, 105, 115, 32, 105, 115, 32, 84, 104, 97, 114, 97, 110, 33]

MESSAGE
         Hola, This is Tharan!
         [72, 111, 108, 97, 44, 32, 84, 104, 105, 115, 32, 105, 115, 32, 84, 104, 97, 114, 97, 110, 33]


ENCRYPTED
         [12474432, 1787605, 7463722, 10725504, 161020, 4340125, 9419594, 1047625, 11623236, 569425, 4340125, 11623236, 569425, 4340125, 9419594, 1047625, 10725504, 2720650, 10725504, 9449714, 5323731]


DECRYPTED
         Hola, This is Tharan!
         [72, 111, 108, 97, 44, 32, 84, 104, 105, 115, 32, 105, 115, 32, 84, 104, 97, 114, 97, 110, 33]
```