Assignment No: 3

Title: Implement RSA Algorithm

Name: Shubhankar jakate
PRN : 22310371
Roll No = 382019

Objective: The key objectives of this assignment are:

1. To study the concept of **asymmetric (public-key) cryptography**.
2. To understand the **RSA algorithm** for encryption and decryption.
3. To implement RSA for secure message communication.

Theory:

Introduction to Asymmetric Cryptography

Unlike symmetric key cryptography (AES, DES) where the same key is used for both encryption and decryption, **asymmetric cryptography** uses **two keys**:

- **Public Key** → Shared openly, used for **encryption**.
- **Private Key** → Kept secret, used for **decryption**.

This removes the problem of secure key distribution that exists in symmetric cryptography.

RSA Algorithm

The **RSA algorithm**, developed in 1977 by **Ron Rivest, Adi Shamir, and Leonard Adleman**, is one of the first public-key cryptosystems and remains widely used today.

- **Type:** Asymmetric (Public Key) Cryptosystem
- **Key Sizes:** 1024, 2048, 4096 bits commonly used
- **Based On:** Difficulty of **factoring large prime numbers**

RSA Key Generation Steps

1. **Choose two large prime numbers** `p` and `q`.
2. **Compute modulus:**

   $n = p \times q$

   (This `n` is part of both public and private keys.)

3. **Compute Euler's Totient Function:**

   $\phi(n)=(p-1)(q-1)$

4. **Choose public exponent `e`:**
   - Must be relatively prime to $\varphi(n)$ (commonly $e = 65537$).
5. **Compute private exponent `d`:**
   - d is the modular inverse of e modulo $\varphi(n)$.

   $$d \times e \equiv 1 \pmod{\phi(n)}$$

   Public Key = (e, n)

   Private Key = (d, n)

RSA Encryption

To encrypt a message M (as a number $< n$):

$$C = M^e \bmod n$$

Where:

- M → plaintext (converted to number)
- C → ciphertext

RSA Decryption

To decrypt:

$$M = C^d \bmod n$$

Where:

- C → ciphertext
- M → original plaintext

Advantages of RSA

- **Secure:** Based on difficulty of factoring large numbers.
- **Key Distribution:** Public key can be freely shared.
- **Digital Signatures:** Provides authentication along with encryption.

Limitations of RSA

- **Slow:** Much slower than symmetric algorithms like AES.
- **Key Size:** Requires very large keys (2048/4096 bits) for strong security.
- **Not used for bulk data:** Instead, RSA is typically used to exchange a symmetric key, which is then used for fast encryption (hybrid cryptography).

Code:

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

def multiplicative_inverse(e, phi):

    d, x1, x2, y1 = 0, 0, 1, 1
    temp_phi = phi
    while e > 0:
        temp1, temp2 = divmod(temp_phi, e)
        temp_phi, e = e, temp2
        x, y = x2 - temp1 * x1, d - temp1 * y1
        x2, x1, d, y1 = x1, x, y1, y
    if temp_phi == 1:
        return d + phi

def generate_keys(p, q):
    n = p * q
    phi = (p - 1) * (q - 1)


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


    d = multiplicative_inverse(e, phi)

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

def encrypt(pk, plaintext):
    key, n = pk
    cipher = [pow(ord(char), key, n) for char in plaintext]
    return cipher
```

```python
def decrypt(pk, ciphertext):
    key, n = pk
    plain = [chr(pow(char, key, n)) for char in ciphertext]
    return ''.join(plain)

if __name__ == "__main__":
    print("Simple RSA Demonstration")


    p = 61
    q = 53

    public, private = generate_keys(p, q)

    print("\nPublic Key:", public)
    print("Private Key:", private)

    message = input("\nEnter a message to encrypt: ")
    encrypted_msg = encrypt(public, message)
    print("Encrypted Message:", encrypted_msg)

    decrypted_msg = decrypt(private, encrypted_msg)
    print("Decrypted Message:", decrypted_msg)
```

OUTPUT :

```
03\Assignment03.py"
Simple RSA Demonstration

Public Key: (7, 3233)
Private Key: (1783, 3233)

Enter a message to encrypt: Hello RSA
Encrypted Message: [1087, 3071, 1877, 1877, 3183, 2774, 1077, 1825, 1317]
Decrypted Message: Hello RSA
PS C:\Users\CHANDRAKANT THAKARE\Desktop\TY IS Assignments>
```