**CNS Exp 6**

**Shashwat Tripathi  
D15A 64**

**Batch C**

**AIM:** Write a program to implement and analyze RSA cryptosystem

**Theory:**

The RSA algorithm is a fundamental and widely used cryptographic system that plays a crucial role in securing digital communication, data transmission, and authentication. It's named after its inventors: Ron Rivest, Adi Shamir, and Leonard Adleman.

At its core, RSA is an asymmetric or public-key cryptography system, meaning it uses a pair of keys for encryption and decryption: a public key and a private key.

**Key Generation:**

The process begins with the generation of these key pairs. To create a pair, two large prime numbers are selected, typically denoted as p and q. These prime numbers are multiplied to obtain a modulus n, which is a fundamental component of both the public and private keys. The modulus n is used to ensure that encryption and decryption operations remain mathematically compatible.

To further strengthen security, the totient (Euler's totient function) of n is calculated as φ(n) = (p - 1)(q - 1). The totient is used to determine the public exponent e, which must be a number coprime (having no common factors other than 1) to φ(n).

Public and Private Keys:

The public key is composed of the modulus n and the public exponent e. This public key is freely shared with anyone who needs to send encrypted messages or verify digital signatures.

The private key consists of the modulus n and the private exponent d. The private key is kept secret and securely stored by its owner.

**Encryption and Decryption:**

When someone wants to send an encrypted message to the owner of the public key:

They convert the message into a numerical format.

The message is then encrypted using the recipient's public key by applying a mathematical operation that involves exponentiation and modulo arithmetic. This produces the ciphertext.

Only the owner of the private key can decrypt this ciphertext back into the original message using the private key. This is done by applying a similar mathematical operation.

**Security:**

RSA's security relies on the inherent difficulty of certain mathematical problems, such as factoring the product of two large prime numbers (n = p \* q). The security strength of RSA increases with the length of the modulus n. Longer keys are considered more secure, but they also require more computational resources to process.

**Applications:**

RSA has numerous applications in the realm of secure communication and data protection. It is used in widely adopted protocols like HTTPS (for secure web browsing), SSH (for secure remote access), and S/MIME (for secure email). RSA also plays a critical role in the creation and verification of digital signatures, ensuring the authenticity and integrity of messages or software.

Code:

Python

import math

def gcd(a, h):

temp = 0

while(1):

temp = a % h

if (temp == 0):

return h

a = h

h = temp

p = 3

q = 7

n = p\*q

e = 2

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

while (e < phi):

# e must be co-prime to phi and

# smaller than phi.

if(gcd(e, phi) == 1):

break

else:

e = e+1

# Private key (d stands for decrypt)

# choosing d such that it satisfies

# d\*e = 1 + k \* totient

k = 2

d = (1 + (k\*phi))/e

# Message to be encrypted

msg = 12.0

print("Message data = ", msg)

# Encryption c = (msg ^ e) % n

c = pow(msg, e)

c = math.fmod(c, n)

print("Encrypted data = ", c)

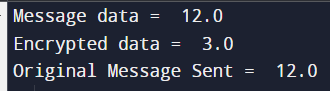
# Decryption m = (c ^ d) % n

m = pow(c, d)

m = math.fmod(m, n)

print("Original Message Sent = ", m)

Output:



**Conclusion:**

Thus, we have implemented and analyzed RSA cryptosystem