CNS PRACTICAL 6

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 $\varphi(n) = (p - 1)(q - 1)$. The totient is used to determine the public exponent e, which must be a number coprime (having no common factors other than 1) to $\varphi(n)$.

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.

1. Primary Use Cases of the RSA Algorithm in Modern Cryptography
Ans:RSA (Rivest–Shamir–Adleman) is an asymmetric cryptographic algorithm used mainly for:

- **Secure Data Encryption:** Encrypting messages so only the intended recipient with the private key can decrypt them.
- **Digital Signatures:** Verifying the authenticity and integrity of digital data by allowing a sender to sign with a private key and recipients to verify with the public key.
- **Key Exchange:** Securely exchanging symmetric keys (e.g., AES keys) over an insecure channel, as in TLS/SSL for HTTPS.
- **Authentication:** Ensuring that a message or user is genuinely from a trusted source.

2. What are the key steps involved in the RSA algorithm?

Ans: The RSA process involves three major phases:

a) Key Generation

- Select two large prime numbers ppp and qqq.
- Compute the modulus n=p×qn = p \times qn=p×q.
- Calculate Euler's totient ϕ(n)=(p−1)(q−1)\phi(n) = (p - 1)(q - 1)ϕ(n)=(p−1)(q−1).
- Choose a public exponent eee such that 1<e<ϕ(n)1 < e < \phi(n)1<e<ϕ(n) and gcd(e,ϕ(n)e, \phi(n)e,ϕ(n)) = 1.
- Compute the private exponent ddd as the modular inverse of eee mod ϕ(n)\phi(n)ϕ(n) (i.e., d×e≡1mod ϕ(n)d \times e \equiv 1 \mod \phi(n)d×e≡1modϕ(n)).

b) Encryption

- Ciphertext CCC is computed as:
  C=Memod nC = M^e \mod nC=Memodn
  where MMM is the plaintext message.

c) Decryption

- Plaintext MMM is recovered as:
  M=Cdmod nM = C^d \mod nM=Cdmodn

3. Explain how the modulus (n) is calculated in RSA. Why is it significant?
Ans:
Calculation:

$n=p\times q$
where p and q are large prime numbers.

Significance:

- **Security Basis:** The strength of RSA relies on the computational difficulty of factoring n back into p and q.
- **Key Component:** Both the public and private keys use the same modulus n.
- **Message Size Limit:** The message M must be a number smaller than n for encryption/decryption.

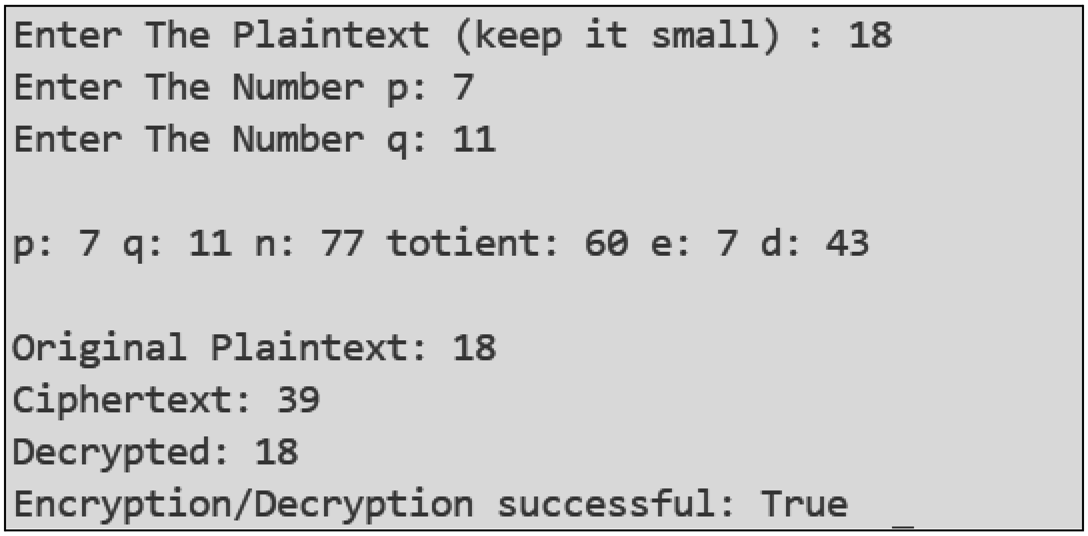
4. Describe a real-world scenario where RSA is used for secure communication.
Ans: RSA is widely used in HTTPS (SSL/TLS) connections:

- When you visit a secure website (`https://`), your browser uses RSA to encrypt a randomly generated symmetric session key (for AES or ChaCha20 encryption).
- Only the server's private key can decrypt this session key.
- After the session key is securely exchanged, all further communication uses faster symmetric encryption, while RSA ensures the initial key exchange is secure.

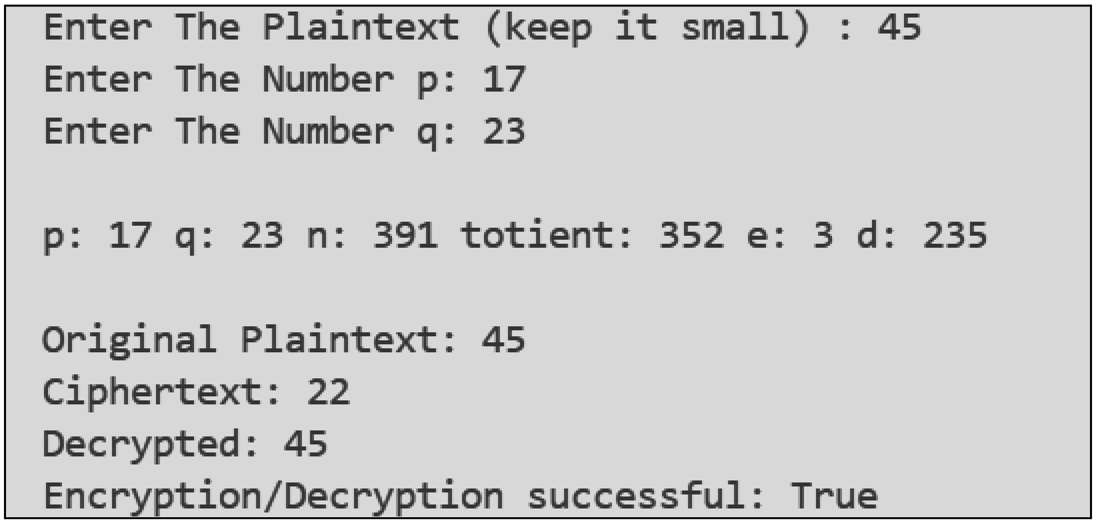
Code:

```
from math import gcd

def extended_gcd(a, b):
    if a == 0:
        return b, 0, 1
    gcd_val, x1, y1 = extended_gcd(b % a, a)
    x = y1 - (b // a) * x1
    y = x1
    return gcd_val, x, y

def mod_inverse(a, m):
    gcd_val, x, y = extended_gcd(a, m)
    if gcd_val != 1:
        return None
    return (x % m + m) % m

plaintext = int(input("Enter The Plaintext (keep it small) : "))
p = int(input("Enter The Number p: "))
q = int(input("Enter The Number q: "))

n = p * q
totient = (p - 1) * (q - 1)

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

d = mod_inverse(e, totient)

print()
print("p:", p, "q:", q, "n:", n, "totient:", totient, "e:", e, "d:", d)
print()

cipher = pow(plaintext, e, n)

decrypted = pow(cipher, d, n)

print("Original Plaintext:", plaintext)
print("Ciphertext:", cipher)
print("Decrypted:", decrypted)
print("Encryption/Decryption successful:", plaintext == decrypted)
```

Output:



```
Enter The Plaintext (keep it small) : 18
Enter The Number p: 7
Enter The Number q: 11

p: 7 q: 11 n: 77 totient: 60 e: 7 d: 43

Original Plaintext: 18
Ciphertext: 39
Decrypted: 18
Encryption/Decryption successful: True  _
```



```
 Enter The Plaintext (keep it small) : 45
 Enter The Number p: 17
 Enter The Number q: 23

 p: 17 q: 23 n: 391 totient: 352 e: 3 d: 235

 Original Plaintext: 45
 Ciphertext: 22
 Decrypted: 45
 Encryption/Decryption successful: True
```