# NETWORK SECURITY

NAME:MUVAA KOMALI

ROLL NO:CS22B1035

COURSE CODE:CS1702

What is RSA?

RSA (Rivest-Shamir-Adleman) is an asymmetric cryptographic algorithm used for secure communication and encryption. It is widely used in secure applications such as SSL/TLS encryption, digital signatures, and secure key exchanges.

Step-by-Step Working of RSA Algorithm

* Key Generation

1)Choose two prime numbers: 𝑝 and q.

2)Compute modulus:

n=p×q

3)Compute Euler’s Totient Function:

ϕ(n)=(p−1)×(q−1)

4)Choose a public exponent e such that:

1<e<ϕ(n)

gcd(𝑒,𝜙(𝑛))=1 (coprime to ϕ(n))

5)Compute the private key d:

d=e −1modϕ(n)

(i.e., 𝑑d is the modular inverse of 𝑒e modulo 𝜙𝑛)

6)Public Key:

(e,n)

7)Private Key:

(d,n)

* Encryption

Convert the plaintext message into numbers (ASCII values if text).

Encrypt each character M using the public key (e,n):

C=modn

The ciphertext (C) is sent to the receiver.

* Decryption

1. The receiver decrypts each encrypted number C using the private key (d,n):

M=modn

1. Convert the numbers back into characters to retrieve the original message.

CODE IMPLEMENTATION:

from sympy import mod\_inverse

# Helper function to compute gcd

def gcd(a, b):

    while b:

        a, b = b, a % b

    return a

# RSA Key Generation

def generate\_keys(p, q):

    if p == q:

        raise ValueError("p and q must be distinct prime numbers.")

    n = p \* q

    phi\_n = (p - 1) \* (q - 1)

    # Choose e such that 1 < e < phi(n) and gcd(e, phi(n)) = 1

    e = 3

    while e < phi\_n and gcd(e, phi\_n) != 1:

        e += 2

    # Compute private key d

    d = mod\_inverse(e, phi\_n)

    return (e, n), (d, n)

# Encrypt a text message

def encrypt\_text(message, public\_key):

    e, n = public\_key

    encrypted\_numbers = [pow(ord(char), e, n) for char in message]  # Convert char to ASCII and encrypt

    return encrypted\_numbers

# Decrypt an encrypted message

def decrypt\_text(ciphertext, private\_key):

    d, n = private\_key

    decrypted\_chars = [chr(pow(num, d, n)) for num in ciphertext]  # Decrypt and convert back to text

    return ''.join(decrypted\_chars)

# User input for prime numbers

p = int(input("Enter prime number p: "))

q = int(input("Enter prime number q: "))

# Generate RSA keys

public\_key, private\_key = generate\_keys(p, q)

# User input for message

message = input("Enter a message (text): ")

# Encrypt and decrypt

ciphertext = encrypt\_text(message, public\_key)

decrypted\_message = decrypt\_text(ciphertext, private\_key)

# Output results

print(f"\nPrime p: {p}")

print(f"Prime q: {q}")

print(f"Public Key (e, n): {public\_key}")

print(f"Private Key d: {private\_key[0]}")

print(f"Original Message: {message}")

print(f"Ciphertext (Encrypted ASCII Values): {ciphertext}")

print(f"Decrypted Message: {decrypted\_message}")

EXPLANATION OF THE CODE:

1.gcd function: Computes the greatest common divisor (GCD) of two numbers, used to check if numbers are coprime.

2. generate\_keys function:

- Takes two prime numbers p and q.

- Calculates n=p×q and ϕ(n)=(p−1)(q−1).

- Chooses an encryption key e such that 1<e<ϕ(n) and gcd(e,ϕ(n))=1.

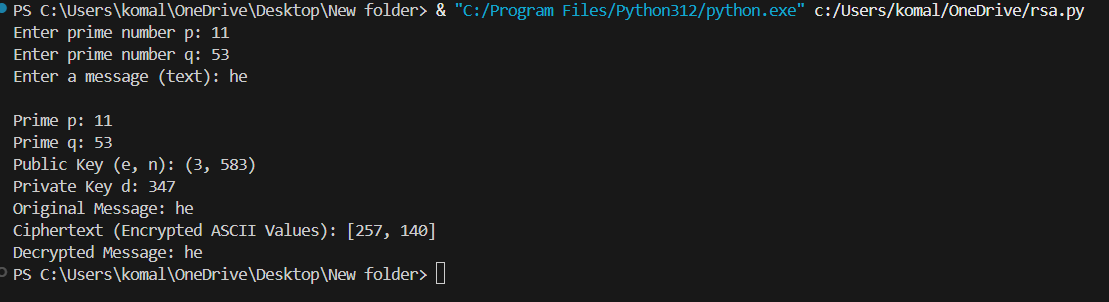
- Finds the private key d using the modular inverse of e

3. encrypt\_text function: Encrypts a message by converting each character into its ASCII value, raising it to the power of e mod n .

4. decrypt\_text function: Decrypts the encrypted message using the private key d to retrieve the original message.

5. The program prompts the user to enter two primes p and q , generates the keys, then encrypts and decrypts a user-provided message, displaying results.

OUTPUT:



1. You chose p = 11 and q = 53, two prime numbers.
2. n = p \* q = 583 is computed, and it's used in both public and private keys.
3. phi(n) = (p-1) \* (q-1) = 520 is calculated.
4. Public key (e, n) is chosen as (3, 583), where e is coprime with phi(n).
5. The private key (d, n) is calculated using the modular inverse of e, which gives d = 347.
6. The message "he" is converted to ASCII values: 'h' = 104, 'e' = 101.
7. Encryption is done by raising each ASCII value to the power of e modulo n:

For 'h': mod 583=257

For 'e': mod583=140

1. The ciphertext is [257, 140].
2. Decryption uses the private key to retrieve the original message.
3. The decrypted message is "he", matching the original.

THANK YOU