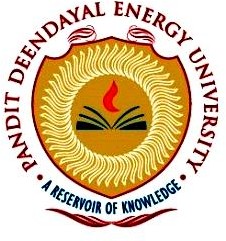
**Information Security** **Lab**

**(20CP304P)**

# **Smit Sutariya**

**Roll no. 21BCP142**

**Div:3 G:5**



**Faculty Name: Hargeet Kaur**

**COMPUTER ENGINEERING**

**School of Technology, Pandit Deendayal Energy University**

**January – 2023**

Experiment: 9

**Aim:** To Study and Implement a program for RSA Algorithm to encrypt and decrypt the message.

Introduction :

The RSA (Rivest-Shamir-Adleman) algorithm is a widely used public-key cryptosystem that was invented by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977. RSA is renowned for its ability to securely encrypt and digitally sign data, making it a fundamental cornerstone of modern information security. The algorithm relies on the mathematical properties of large prime numbers and modular arithmetic to provide secure communication and data protection. RSA has applications in secure email, e-commerce, digital signatures, and more.

In RSA, each user has a pair of keys: a public key, which is known to everyone, and a private key, which is kept secret. The public key is used for encryption, while the private key is used for decryption. The security of RSA is grounded in the difficulty of factoring the product of two large prime numbers, which is an essential component of the algorithm. As prime factorization is a computationally intensive task, RSA encryption remains secure as long as sufficiently large keys are used.

One of the key advantages of RSA is its versatility in providing both confidentiality and authenticity in digital communication. It ensures that only the intended recipient can decrypt the message with their private key, while the sender can digitally sign their messages with their private key to prove their identity. RSA's enduring popularity, based on the principles of asymmetric cryptography, has made it a vital tool in the protection of sensitive information and the establishment of secure online transactions in our increasingly digital world.

**Code:**

import random

def is\_prime(num):

    if num <= 1:

        return False

    if num <= 3:

        return True

    if num % 2 == 0 or num % 3 == 0:

        return False

    i = 5

    while i \* i <= num:

        if num % i == 0 or num % (i + 2) == 0:

            return False

        i += 6

    return True

def gcd(a, b):

    while b:

        a, b = b, a % b

    return a

def mod\_inverse(a, m):

    m0, x0, x1 = m, 0, 1

    while a > 1:

        q = a // m

        m, a = a % m, m

        x0, x1 = x1 - q \* x0, x0

    return x1 + m0 if x1 < 0 else x1

def list\_prime(num):

    prime\_list = []

    for i in range(2, num):

        if is\_prime(i):

            prime\_list.append(i)

    return prime\_list

def generate\_prime(bits):

    num = random.getrandbits(bits)

    return prime\_number\_list[num % len(prime\_number\_list)]

# Generate public and private keys

def generate\_keys(bits):

    p = generate\_prime(bits)

    print(f"p={p}")

    q = generate\_prime(bits)

    print(f"q={q}")

    n = p \* q

    phi = (p - 1) \* (q - 1)

    e = 65537

    d = mod\_inverse(e, phi)

    return (e, n), (d, n)

# Encrypt a message

def encrypt(public\_key, message):

    e, n = public\_key

    encrypted = [pow(ord(char), e, n) for char in message]

    return encrypted

# Decrypt a message

def decrypt(private\_key, encrypted):

    d, n = private\_key

    decrypted = [chr(pow(char, d, n)) for char in encrypted]

    return ''.join(decrypted)

if \_\_name\_\_ == '\_\_main\_\_':

    prime\_number\_list = list\_prime(10000)

    public\_key, private\_key = generate\_keys(1024)

    message = "Hello smith"

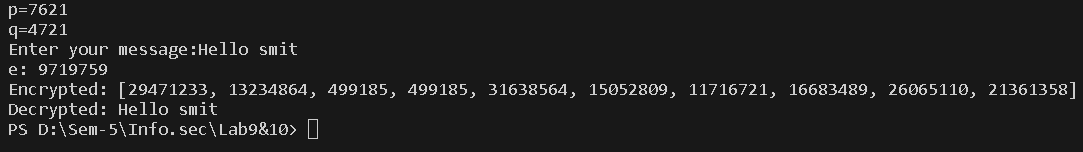
    encrypted\_message = encrypt(public\_key, message)

    print("Encrypted:", encrypted\_message)

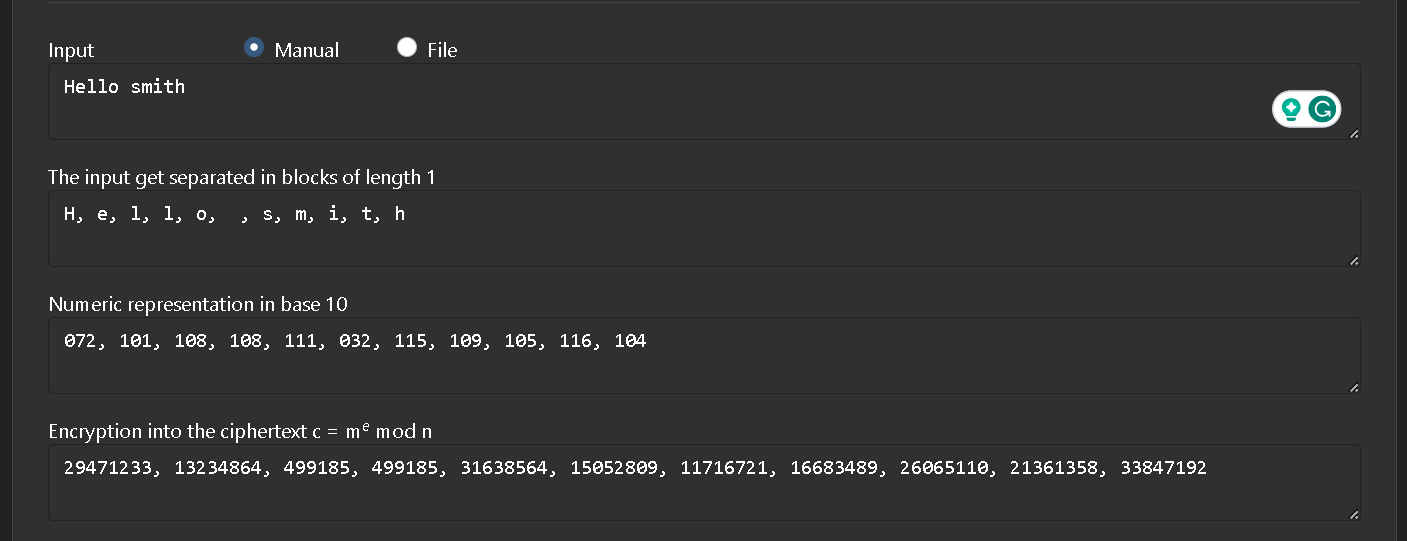
    decrypted\_message = decrypt(private\_key, encrypted\_message)

    print("Decrypted:", decrypted\_message)

**OUTPUT:-**

****

**OUTPUT(CRYPTOOL):**

**CRYPTANALYSIS:**

Cryptanalysis of the RSA algorithm involves attempting to break its security by exploiting weaknesses in the underlying mathematics or implementation. While RSA is considered a robust encryption method, there are several cryptanalysis techniques and potential vulnerabilities to consider:

**Integer Factorization:** RSA's security relies on the difficulty of factoring the product of two large prime numbers. Cryptanalysts continually work on more efficient factoring algorithms. The development of powerful quantum computers poses a potential threat to RSA, as they may be able to factor large numbers much faster than classical computers.

**Timing Attacks:** Cryptanalysts can exploit information leakage through the timing of encryption or decryption operations. By analyzing the time taken to execute these operations, attackers might gain insights into the private key.

**Chosen Ciphertext Attacks (CCA)**: In CCA attacks, an attacker can interact with an oracle to decrypt chosen ciphertexts. While modern RSA implementations incorporate padding schemes like OAEP or PKCS#1 v1.5 to mitigate these attacks, vulnerabilities may still arise if these schemes are not correctly implemented.

**Side-Channel Attacks:** RSA implementations can be susceptible to side-channel attacks based on the electromagnetic emissions, power consumption, or other observable aspects of the hardware or software during cryptographic operations. Attackers can use these observable signals to deduce private key information.

**Weak Key Generation:** Weak key generation can lead to vulnerabilities in RSA. If users generate keys with insufficient randomness or use predictable prime numbers, it becomes easier for cryptanalysts to crack the encryption. Secure key generation practices are crucial to RSA's strength.

**APPLICATIONS:**

The RSA (Rivest-Shamir-Adleman) algorithm is a versatile encryption and digital signature scheme with various applications in information security. Here are six prominent applications of the RSA algorithm:

**Secure Communication:** RSA is widely used in securing communication over the internet, particularly for encrypting email and instant messaging. It ensures that data exchanged between parties remains confidential and cannot be intercepted or deciphered by unauthorized individuals.

**Digital Signatures:** RSA is employed for creating digital signatures to verify the authenticity and integrity of digital documents, software, or messages. By signing data with a private key, a sender can prove that the content has not been tampered with and is genuinely from them.

**Secure Web Browsing:** RSA is a fundamental component of secure web browsing. It is used to establish SSL/TLS (Secure Sockets Layer/Transport Layer Security) connections, encrypting data exchanged between web browsers and servers. This ensures the privacy of sensitive information like credit card details during online transactions.

**Authentication and Access Control:** RSA is employed for user authentication in various systems, including VPNs, remote access, and secure login processes. Users have their private keys to prove their identity, which enhances security in access control.

**REFERENCES:**

**Stallings, W. (2017). *Cryptography and Network Security: Principles and Practice.*** Pearson. <https://www.pearson.com/en-us/subject-catalog/p/cryptography-and-network-security-principles-and-practice/P200000003477>