**Cryptography and Network Security Lab**

**Practical 08**

Q. Implement Rivest-Shamir-Adleman (RSA) Algorithm.

**Theory:**

RSA (Rivest–Shamir–Adleman) is a widely used public-key cryptosystem for secure data transmission and digital signatures. It was invented by Ron Rivest, Adi Shamir, and Leonard Adleman in 1977. RSA relies on the mathematical properties of large prime numbers and modular arithmetic. Here's an overview of how RSA works:

1. Key Generation:

- Choose two large prime numbers, p and q.

- Compute their product, n = p \* q. n is used as the modulus for both the public and private keys.

- Compute the totient (Euler's totient function) of n, φ(n) = (p-1) \* (q-1).

- Choose an integer, e, such that 1 < e < φ(n), and e is coprime to φ(n). This is the public exponent.

- Calculate the private exponent, d, which is the modular multiplicative inverse of e modulo φ(n), i.e., (d \* e) % φ(n) = 1.

The public key consists of (n, e), and the private key is (n, d).

2. Encryption (By Sender):

To encrypt a message (plaintext) M, the sender performs the following:

- Represent the message as a number m such that 0 <= m < n.

- Compute the ciphertext C = M^e mod n.

- Send C to the recipient.

3. Decryption (By Receiver):

The receiver uses the private key (n, d) to decrypt the ciphertext C:

- Compute the plaintext message M = C^d mod n.

The security of RSA relies on the difficulty of factoring the modulus n (given only n and e) to determine p and q, which would allow someone to calculate d. If the key length (i.e., the bit length of n) is large enough, RSA is considered secure.

Key points about RSA:

- RSA is asymmetric, meaning there are separate keys for encryption (public key) and decryption (private key).

- Public keys are used for encryption and digital signatures, while private keys are used for decryption and creating digital signatures.

- The security of RSA depends on the difficulty of factoring the modulus n.

- RSA key length (in bits) affects the security level. Longer keys provide stronger security but also require more computational resources.

RSA is commonly used in securing communication over the internet, including secure email, web browsing, and more. It's also used for digital signatures to verify the authenticity and integrity of messages and data.

**Code:**

#include <iostream>

#include <string>

#include <vector>

#include <cmath>

#include <random>

using namespace std;

*// Function for extended Euclidean Algorithm*

int ansS, ansT;

int findGcdExtended(int *r1*, int *r2*, int *s1*, int *s2*, int *t1*, int *t2*) {

    int s = 0; *// Declare 's' here*

    int t = 0; *// Declare 't' here*

    if (*r2* == 0) {

        ansS = *s1*;

        ansT = *t1*;

        return *r1*;

    }

    int q = *r1* / *r2*;

    int r = *r1* % *r2*;

    s = *s1* - q \* *s2*;

    t = *t1* - q \* *t2*;

    return findGcdExtended(*r2*, r, *s2*, s, *t2*, t);

}

int modInverse(int *A*, int *M*) {

    int x, y;

    int g = findGcdExtended(*A*, *M*, 1, 0, 0, 1);

    if (g != 1) {

        cout << "Inverse doesn't exist";

        return 0;

    } else {

        int res = (ansS % *M* + *M*) % *M*;

        return res;

    }

}

long long powM(long long *a*, long long *b*, long long *n*) {

    if (*b* == 1)

        return *a* % *n*;

    long long x = powM(*a*, *b* / 2, *n*);

    x = (x \* x) % *n*;

    if (*b* % 2)

        x = (x \* *a*) % *n*;

    return x;

}

int findGCD(int *num1*, int *num2*) {

    if (*num1* == 0)

        return *num2*;

    return findGCD(*num2* % *num1*, *num1*);

}

*// Function to convert text to a numerical representation*

vector<int> textToNumbers(const string& *text*) {

    vector<int> numbers;

    for (char c : *text*) {

        numbers.push\_back(static\_cast<int>(c));

    }

    return numbers;

}

*// Function to convert numerical representation to text*

string numbersToText(const vector<int>& *numbers*) {

    string text;

    for (int num : *numbers*) {

        text.push\_back(static\_cast<char>(num));

    }

    return text;

}

*// Code to demonstrate RSA algorithm for text*

int main() {

*// Two random prime numbers (modify these)*

    long long p, q;

    cout << "Enter two prime numbers (p and q): ";

    cin >> p >> q;

*// First part of public key: n = p \* q*

    long long n = p \* q;

*// Finding other part of public key (e stands for encrypt)*

    long long e;

    cout << "Enter the public exponent (e): ";

    cin >> e;

*// Private key (d stands for decrypt)*

    long long d = modInverse(e, (p - 1) \* (q - 1));

*// Display public and private keys*

    cout << "Public key: <" << e << "," << n << ">" << endl;

    cout << "Private key: <" << d << "," << n << ">" << endl;

*// Message to be encrypted*

    string message;

    cout << "Enter the message to encrypt: ";

    cin.ignore(); *// Ignore any previous newline character*

    getline(cin, message);

*// Convert message to numbers*

    vector<int> numbers = textToNumbers(message);

*// Encrypt the message*

    vector<int> encrypted\_message;

    for (int num : numbers) {

        long long c = powM(num, e, n);

        encrypted\_message.push\_back(static\_cast<int>(c));

    }

    cout << "Encrypted Message: ";

    for (int num : encrypted\_message) {

        cout << num << " ";

    }

    cout << endl;

*// Decrypt the message*

    vector<int> decrypted\_numbers;

    for (int num : encrypted\_message) {

        long long m = powM(num, d, n);

        decrypted\_numbers.push\_back(static\_cast<int>(m));

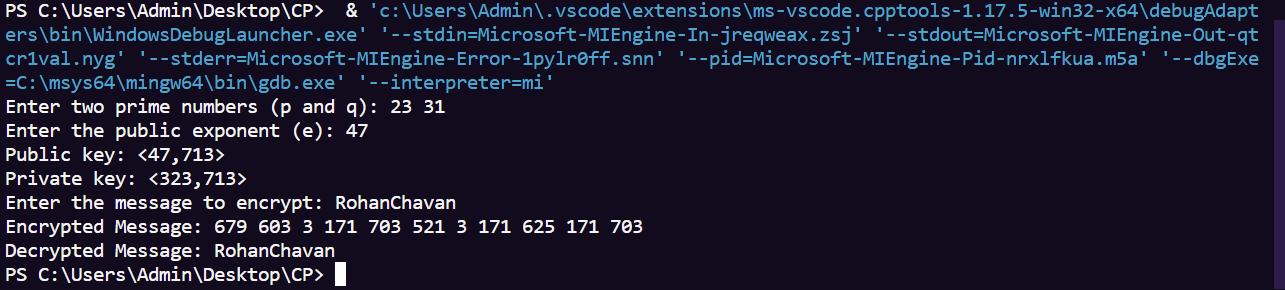
    }

    string decrypted\_message = numbersToText(decrypted\_numbers);

    cout << "Decrypted Message: " << decrypted\_message << endl;

    return 0;

}

Results: