**PRACTICAL-8**

**AIM: Write a C program to implement RSA encryption and decryption algorithm.**

**INTRODUCTION:**

* RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption. It was invented by Rivest, Shamir and Adleman in year 1978 and hence name **RSA** algorithm.
* The RSA algorithm holds the following features –
* RSA algorithm is a popular exponentiation in a finite field over integers including prime numbers.
* The integers used by this method are sufficiently large making it difficult to solve.
* There are two sets of keys in this algorithm: private key and public key.
* You will have to go through the following steps to work on RSA algorithm –

**Step 1: Generate the RSA modulus**

* The initial procedure begins with selection of two prime numbers namely p and q, and then calculating their product N, as shown −
* N=p\*q
* Here, let N be the specified large number.

**Step 2: Derived Number (e)**

* Consider number e as a derived number which should be greater than 1 and less than (p-1) and (q-1). The primary condition will be that there should be no common factor of (p-1) and (q-1) except 1.

**Step 3: Public key**

* The specified pair of numbers **n** and **e** forms the RSA public key and it is made public.

**Step 4: Private Key**

* Private Key **d** is calculated from the numbers p, q and e. The mathematical relationship between the numbers is as follows −
* ed = 1 mod (p-1) (q-1)
* The above formula is the basic formula for Extended Euclidean Algorithm, which takes p and q as the input parameters.

## Encryption Formula

* Consider a sender who sends the plain text message to someone whose public key is **(n,e).** To encrypt the plain text message in the given scenario, use the following syntax −
* C = Pe mod n

## Decryption Formula

* The decryption process is very straightforward and includes analytics for calculation in a systematic approach. Considering receiver**C** has the private key **d**, the result modulus will be calculated as −
* Plaintext = Cd mod n

**CODE**:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

int checkPrime(int n) {

int i;

int m = n / 2;

for (i = 2; i <= m; i++) {

if (n % i == 0) {

return 0; // Not Prime

}

}

return 1; // Prime }

int findGCD(int n1, int n2) {

int i, gcd;

for(i = 1; i <= n1 && i <= n2; ++i) {

if(n1 % i == 0 && n2 % i == 0)

gcd = i;

}

return gcd; }

int powMod(int a, int b, int n) {

long long x = 1, y = a;

while (b > 0) {

if (b % 2 == 1)

x = (x \* y) % n;

y = (y \* y) % n; // Squaring the base

b /= 2;

}

return x % n; }

int main(int argc, char\* argv[]) {

int p, q;

int n, phin;

int data, cipher, decrypt;

while (1) {

printf("Enter any two prime numbers: ");

scanf("%d %d", &p, &q);

if (!(checkPrime(p) && checkPrime(q)))

printf("Both numbers are not prime. Please enter prime numbers only...\n");

else if (!checkPrime(p))

printf("The first prime number you entered is not prime, please try again...\n");

else if (!checkPrime(q))

printf("The second prime number you entered is not prime, please try again...\n");

else

break; }

n = p \* q;

phin = (p - 1) \* (q - 1);

int e = 0;

for (e = 5; e <= 100; e++) {

if (findGCD(phin, e) == 1)

break; }

int d = 0;

for (d = e + 1; d <= 100; d++) {

if ( ((d \* e) % phin) == 1)

break; }

printf("Value of e: %d\nValue of d: %d\n", e, d);

printf("Enter some numerical data: ");

scanf("%d", &data);

cipher = powMod(data, e, n);

printf("The cipher text is: %d\n", cipher);

decrypt = powMod(cipher, d, n);

printf("The decrypted text is: %d\n", decrypt);

return 0; }

**OUTPUT**:

