Experiment no.: 06

Title: Encrypt & decrypt a plaintext using RSA algorithm.
Course Outcome: Use cryptography algorithms and protocols to achieve Computer Security
Theory:
RSA Algorithm in Cryptography:
RSA(Rivest-Shamir-Adleman) Algorithm is an asymmetric or public-key cryptography algorithm which means it works on two different keys: Public Key and Private Key. The Public Key is used for encryption and is known to everyone, while the Private Key is used for decryption and must be kept secret by the receiver. RSA Algorithm is named after Ron Rivest, Adi Shamir and Leonard Adleman, who published the algorithm in 1977.

Example of Asymmetric Cryptography:
If Person A wants to send a message securely to Person B:

- Person A encrypts the message using Person B's Public Key.
- Person B decrypts the message using their Private Key.

RSA Algorithm:
RSA Algorithm is based on factorization of large number and modular arithmetic for encrypting and decrypting data.
It consists of three main stages:
Key Generation: Creating Public and Private Keys
Encryption: Sender encrypts the data using Public Key to get cipher text.
Decryption: Decrypting the cipher text using Private Key to get the original data.

1. Key Generation:
Choose two large prime numbers, say p and q. These prime numbers should be kept secret.
Calculate the product of primes, $n = p * q$. This product is part of the public as well as the private key.
Calculate Euler Totient Function$\Phi(n)$ as $\Phi(n) = \Phi(p * q) = \Phi(p) * \Phi(q) = (p - 1) * (q - 1)$.
Choose encryption exponent e, such that
$1 < e < \Phi(n)$, and
$\gcd(e, \Phi(n)) = 1$, that is e should be co-prime with $\Phi(n)$.
Calculate decryption exponent d, such that

(d * e) ≡ 1 mod Φ(n), that is d is modular multiplicative inverse of e mod Φ(n).
Some common methods to calculate multiplicative inverse are: Extended Euclidean Algorithm, Fermat's Little Theorem, etc.
We can have multiple values of d satisfying (d * e) ≡ 1 mod Φ(n) but it does not matter which value we choose as all of them are valid keys and will result into same message on decryption.
Finally, the Public Key = (n, e) and the Private Key = (n, d).

2. Encryption:
To encrypt a message M, it is first converted to numerical representation using ASCII and other encoding schemes. Now, use the public key (n, e) to encrypt the message and get the cipher text using the formula:
C = Me mod n, where C is the Cipher text and e and n are parts of public key.

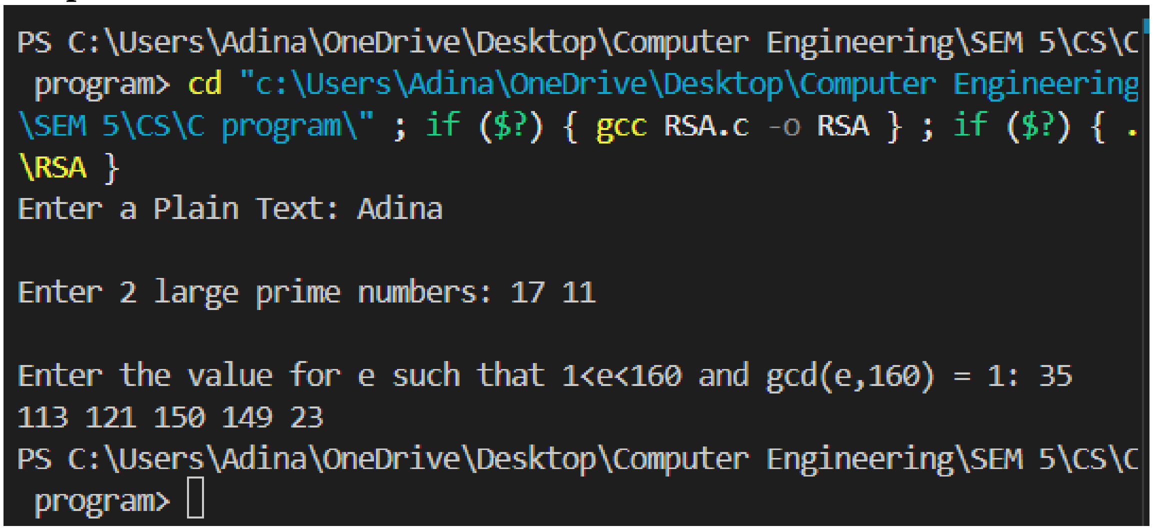
3. Decryption:
To decrypt the cipher text C, use the private key (n, d) and get the original data using the formula:
M = Cd mod n, where M is the message and d and n are parts of private key.

Code:

```
#include <stdio.h>
#include <string.h>
#include <math.h>
int main()
{
   int p, q, e;
   char pt[100];
   int c[100];
   printf("Enter a Plain Text:");
   scanf("%s", pt);
   printf("\nEnter 2 large prime numbers:");
   scanf("%d %d", &p, &q);
   int n = p * q;
   int phi = (p - 1) * (q - 1);
   printf("\nEnter the value for e such that 1<e<%d and gcd(e,%d) = 1: ", phi,
phi);
   scanf("%d", &e);
   int len = strlen(pt);
   for (int i = 0; i < len; i++)
```

```
    {
        int m = (int)pt[i];
        double power_val = pow(m, e);
        double mod_val = fmod(power_val, n);
        c[i] = (int)mod_val;
        printf("%d ", c[i]);
    }
    printf("\n");
    return 0;
}
```

Output:



```
PS C:\Users\Adina\OneDrive\Desktop\Computer Engineering\SEM 5\CS\C
 program> cd "c:\Users\Adina\OneDrive\Desktop\Computer Engineering
\SEM 5\CS\C program\" ; if ($?) { gcc RSA.c -o RSA } ; if ($?) { .
\RSA }
Enter a Plain Text: Adina

Enter 2 large prime numbers: 17 11

Enter the value for e such that 1<e<160 and gcd(e,160) = 1: 35
113 121 150 149 23
PS C:\Users\Adina\OneDrive\Desktop\Computer Engineering\SEM 5\CS\C
 program> 
```

Conclusion:
I have successfully written C program to Encrypt a plaintext using RSA algorithm.