Experiment 8

Aim: Write a program to implement encryption and decryption using the RSA algorithm.

Theory: RSA is a cryptosystem for public-key encryption, and is widely used for securing sensitive data, particularly when being sent over an insecure network such as the Internet. Public-key cryptography, also known as asymmetric cryptography, uses two different but mathematically linked keys, one public and one private. The public key can be shared with everyone, whereas the private key must be kept secret.
Many protocols like SSH, OpenPGP, S/MIME, and SSL/TLS rely on RSA for encryption and digital signature functions. It is also used in software programs -- browsers are an obvious example, which need to establish a secure connection over an insecure network like the Internet or validate a digital signature. RSA signature verification is one of the most commonly performed operations in IT.
Explaining RSA's popularity RSA derives its security from the difficulty of factoring large integers that are the product of two large prime numbers. Multiplying these two numbers is easy, but determining the original prime numbers from the total -- factoring -- is considered infeasible due to the time it would take even using today's supercomputers.

The public and the private key-generation algorithm is the most complex part of RSA cryptography. Two large prime numbers, p and q, are generated using the Rabin-Miller primality test algorithm. A modulus n is calculated by multiplying p and q. This number is used by both the public and private keys and provides the link between them. Its length, usually expressed in bits, is called the key length. The public key consists of the modulus n, and a public exponent, e, which is normally set at 65537, as it's a prime number that is not too large. The e figure doesn't have to be a secretly selected prime number as the public key is shared with everyone. The private key consists of the modulus n and the private exponent d, which is calculated using the Extended Euclidean algorithm to find the multiplicative inverse with respect to the totient of n.

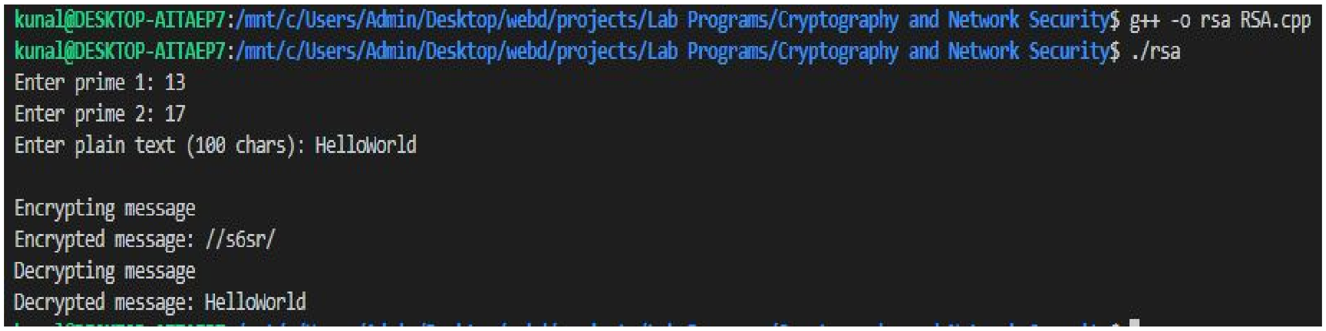
- Encryption
  Sender A does the following: -
  1. Obtains the recipient B's public key (n,e).
  2. Represents the plaintext message as a positive integer m with $1<m<n$.
  3. Computes the ciphertext c = (me mod n) and Sends the ciphertext c to B.
- Decryption
  Recipient B does the following: -
  1. Uses his private key (n,d) to compute m=cd mod n.
  2. Extracts the plaintext from the message representative m.

Source Code:

```
#include<iostream>
#include<cstdio>
#include<math.h>
#include<string.h>
#include<stdlib.h>
using namespace std;
int p, q, n, t, flag, e[100], d[100], temp[100], j, m[100],
en[100], i;
char msg[100];
int prime(long int pr)
        {
        int i;
        j = sqrt(pr);
        for (i = 2; i <= j; i++)
        {if (pr % i == 0)
        return 0;}
        return 1;
        }
void ce()
        {
        int k;
        k = 0;
        for (i = 2; i < t; i++)
        {if (t % i == 0)
        continue;
        flag = prime(i);
        if (flag == 1 && i != p && i != q)
        {e[k] = i;
        flag = cd(e[k]);
        if (flag > 0)
        {d[k] = flag;
          k++;}
        if (k == 99)
        break;
        }}
        }
        long int cd(long int x)
        {long int k = 1;
        while (1)
        {
        k = k + t;
        if (k % x == 0)
        return (k / x);}
        }
void encrypt()
        {
        long int pt, ct, key = e[0], k, len;
        i = 0;
        len = strlen(msg);
        while (i != len)
        {pt = m[i];
        pt = pt - 96;
        k = 1;
        for (j = 0; j < key; j++)
        {k = k * pt;
        k = k % n;
        }
        temp[i] = k;
        ct = k + 96;
        en[i] = ct;
        i++;
        }
        en[i] = -1;
        cout << "Encrypted message: ";
        for (i = 0; en[i] != -1; i++)
        printf("%c", en[i]);
        }
void decrypt()
        {
        long int pt, ct, key = d[0], k;
        i = 0;
        while (en[i] != -1)
        {ct = temp[i];
        k = 1;
        for (j = 0; j < key; j++)
        {k = k * ct;
        k = k % n;}
```

```
          pt = k + 96;
          m[i] = pt;
          i++;
          }
          m[i] = -1;
          cout << "Decrypted message: ";
          for (i = 0; m[i] != -1; i++)
          printf("%c", m[i]);
          }
int main()
          {
          cout << "Enter prime 1: ";
          cin >> p;
          flag = prime(p);
          if (flag == 0)
          {cout << "Entered Value is not PRIME Exiting";
          exit(1);}
          cout << "Enter prime 2: ";
          cin >> q;
          flag = prime(q);
          if (flag == 0 || p == q)
          {cout << "Entered Value is not PRIME Exiting";
          exit(1);}
          cout << "Enter plain text (100 chars): ";
          fflush(stdin);
          cin >> msg;
          for (i = 0; msg[i] != '\0'; i++)
          m[i] = msg[i];
          n = p * q;
          t = (p - 1) * (q - 1);
          ce();
          cout << endl << "Encrypting message" << endl;
          encrypt();
          cout << endl << "Decrypting message" << endl;
          decrypt();
          cout << endl;
          return 0;
          }
```

Output:



```
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab Programs/Cryptography and Network Security$ g++ -o rsa RSA.cpp
kunal@DESKTOP-AITAEP7:/mnt/c/Users/Admin/Desktop/webd/projects/Lab Programs/Cryptography and Network Security$ ./rsa
Enter prime 1: 13
Enter prime 2: 17
Enter plain text (100 chars): HelloWorld

Encrypting message
Encrypted message: //s6sr/
Decrypting message
Decrypted message: HelloWorld
```

Learning Outcomes:

In RSA cryptography, both the public and the private keys can encrypt a message; the opposite key from the one used to encrypt a message is used to decrypt it. This attribute is one reason why RSA has become the most widely used asymmetric algorithm.