Name : Omkar Gurav
Roll no : 8048

Assignment No: 2

Title: Write a program in C++ or JAVA to implement RSA algorithm for key generation and Cipher verification.

Objective: To study,
1. Public key algorithm.
2. RSA algorithm
3. Concept of Public key and Private Key.

Theory:

Public Key Algorithm:

Asymmetric algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristics:

- It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key.

In addition, some algorithms, such as RSA, also exhibit the following characteristics:

- Either of the two related keys can be used for encryption, with the other used for decryption.

Public key encryption scheme has six ingredients:

- **Plaintext:** This is readable message or data that is fed into the algorithm as input.
- **Encryption algorithm:** The encryption algorithm performs various transformations on the plaintext.
- **Public and private key:** This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public or private key that is provided as input.
- **Cipher text:** This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different cipher texts.
- **Decryption algorithm:** This algorithm accepts the cipher text and the matching key and produces the original plaintext.

The essential steps are as the following:

1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or the other accessible file. This is the public key. The companion key is kept private. As figure suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, the decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's private key.

The RSA Algorithm:

The scheme developed by Rivest, Shamir and Adleman makes use of an expression with exponentials. Plaintext is encrypted in blocks, with each block having a binary value less than some number n. That is the block size must be less than or equal to log2 (n); in practice the block size is I bits, where $2i<n<=2i+1$. Encryption and decryption are of the following form, for some plaintext block M and cipher text block C:

$C = M^e \bmod n$

$M = C^d \bmod n = (M^e)^d \bmod n = M^{ed} \bmod n$

Both sender and receiver must know the value of n. The sender knows the value of e, and only the receiver knows the value of d. Thus, this is a public-key encryption algorithm with a public key of PU = {e, n} and a private key of PR = {d, n}. For this algorithm to be satisfactory for public key encryption, the following requirements must meet:

1. It possible to find values of e, d, n such that Med mod n = M for all M<n.
2. It is relatively easy to calculate $M^e \bmod n$ and $C^d \bmod n$ for all values of M<n.
3. It is feasible to determine d given e and n.

Figure: The RSA Algorithm

Example:

1. Select two prime numbers, p = 17 and q = 11.
2. Calculate n = pq = 17*11 = 187.
3. Calculate Ø (n) = (p-1)(q-1) = 16*10 = 160.
4. Select e such that relatively prime to Ø (n) = 160 and less than Ø (n); we choose e = 3.
5. Determine d such that de ≡ 1 (mod 160) and d < 160. The correct value is d = 107, because 107*3 = 321 = 160*2+1; d can be calculated using the extended Euclid's algorithm.

The resulting keys are public key {e, n} = {3, 187} and private key {d, n} = {107, 187}. The example shows the use of these keys for plaintext input of M=88.

Advantages:

1. Easy to implement.

Disadvantages:

1. Anyone can announce the public key.

Input:

- Two prime numbers p = 17 and q = 11.
- Plain text = 88.

Output:

- Public key (e, n) : (3, 187)
- Private key (d, n) : (107,187)
- Encrypted text = 11
- Decrypted text = 88

•

Algorithm:

1. Start
2. Input two prime numbers p and q and check if both of them are prime.
3. Calculate n = pq.
4. Calculate Ø (n) = (p-1)(q-1).
5. Show n and phi_n.
6. Take input plaintext and check if it is less than n.
7. Calculate e.
8. Calculate d.
9. Determine public key and private key.
10. Show public key and private key
11. Encrypt the plaintext and show encrypted text.
12. Decrypt the encrypted text and show decrypted text.
13. Stop.

Conclusion: We have studied and implemented the RSA - Public key algorithm.

RSA.cpp

```
#include<iostream>
#include<math.h>
using namespace std;


class RSA
{
   public:
```

```
    double p, q, n, phi_n;
    double d, e, plain_text, encrypted_text, decrypted_text;



    void get_data();
    static bool check_prime_number(int a);
    static int gcd(int a, int b);
    void calculate_e();
    void calculate_d();
    void encrypt();
    void decrypt();
    static int mod(int a,int b,int c);

};



int RSA::mod(int a, int b, int c)
{
    // Using (ab % c) = ( (a % c)(b % c) ) % c

    int res = 1;

    a = a % c;

    if (a == 0)
      return 0;

    while (b > 0)
```

```
    {
        if (b%2 == 1)
            res = (res*a) % c;

        b = b/2;
        a = (a*a) % c;
    }

    return res;
}


bool RSA::check_prime_number(int a)
{

   if(a == 2)
   {
     return true;
   }
   else if( (a % 2 == 0) || (a == 1) )
   {
       return false;
   }

   for(int i = 3; i < sqrt(a) + 1; i = i+2)
      if(a % i == 0)
        return false;
```

```
   return true;

}


void RSA::get_data()
{
   int flag = 1;


   while(flag)
   {

        cout << "\nEnter two prime numbers : ";
        cin >> p >> q;

        if( (check_prime_number(p)) && (check_prime_number(q)) )
        {
           n = p*q;
           cout << "\nn : " << n;

           phi_n = (p-1)*(q-1);
           cout << "\nphi_n : " << phi_n;

           while(flag)
           {
              cout << "\n\nEnter plain text : ";
              cin >> plain_text;
```

```
            if(plain_text >= n)
            {
               cout << "\n\nPlain text should be smaller than n !!! Enter again.";
            }
            else
              flag = 0;

          }

          flag = 0;

        }
        else
        {

        cout << "\nNumbers are not prime !!! Enter again.";

        }

    }

}


int RSA::gcd(int a, int b)
{
   if (b == 0)
```

```
    return a;

    return gcd(b, a%b);
}



void RSA::calculate_e()
{

    e = 2;

    while(e < phi_n)
    {
      if((gcd(e,phi_n) == 1))
        break;
      else
        e++;
    }

    cout << "\nPublic key (e,n) : (" << e << "," << n << ")" ;

}



void RSA::calculate_d()
{

    for(int i=1; i < phi_n; i++)
```

```
    {
        if(( (int)e*i) % (int)phi_n == 1)
          d = i;
    }

    cout << "\nPrivate key (d,n) : (" << d << "," << n << ")" ;

}


void RSA::encrypt()
{

  encrypted_text = mod(plain_text,e,n);

  cout << "\n\nEncrypted text : " << encrypted_text;

}


void RSA::decrypt()
{

  decrypted_text = mod(encrypted_text,d,n);
  cout << "\n\nDecrypted text : " << decrypted_text;

}
```

```
int main()
{
   RSA r;
   r.get_data();
   r.calculate_e();
   r.calculate_d();
   r.encrypt();
   r.decrypt();

}
```

Output :

```
Enter two prime numbers : 43
71

n : 3053
phi_n : 2940

Enter plain text : 34

Public key (e,n) : (11,3053)
Private key (d,n) : (1871,3053)

Encrypted text : 949
```

Decrypted text : 34