Roll No: 2103163
Batch: C32
Name: Om Shete

Experiment No 5

Aim: Implementation of RSA cryptosystem.

Description:

The RSA algorithm is an asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes the Public Key is given to everyone and the Private key is kept private.

An example of asymmetric cryptography:

A client (for example browser) sends its public key to the server and requests some data.
The server encrypts the data using the client's public key and sends the encrypted data.
The client receives this data and decrypts it.
Since this is asymmetric, nobody else except the browser can decrypt the data even if a third party has the public key of the browser.

The idea! The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is a multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, the private key is compromised. Therefore encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024-bit keys could be broken in the near future. But till now it seems to be an infeasible task.

Let us learn the mechanism behind the RSA algorithm : >> Generating Public Key:

Select two prime no's. Suppose P = 53 and Q = 59.
Now First part of the Public key : n = P*Q = 3127.
We also need a small exponent say e :
But e Must be
An integer.
Not be a factor of Φ(n).
1 < e < Φ(n) [Φ(n) is discussed below],
Let us now consider it to be equal to 3.
Our Public Key is made of n and e

>> Generating Private Key:

Roll No: 2103163
Batch: C32
Name: Om Shete

We need to calculate Φ(n) :
Such that Φ(n) = (P-1)(Q-1)
so, Φ(n) = 3016
Now calculate Private Key, d :
d = (k*Φ(n) + 1) / e for some integer k
For k = 2, value of d is 2011.

Now we are ready with our – Public Key (n = 3127 and e = 3) and Private Key(d = 2011) Now we will encrypt “HI”:

Convert letters to numbers : H = 8 and I = 9
Thus Encrypted Data c = (89e)mod n
Thus our Encrypted Data comes out to be 1394
Now we will decrypt 1394 :
Decrypted Data = (cd)mod n
Thus our Encrypted Data comes out to be 89
8 = H and I = 9 i.e. "HI".

Code:

```
#include <iostream>
using namespace std;

int power(int a, int b, int mod) {
  int x = a;
  int result = 1;
  int count = 0;
  cout << "------------------------------------------------" << endl;
  while (b > 0) {
    if ((b & 1) == 1)
      result = (result * a) % mod;
    if ((b & 1) == 1)
      cout << "Binary Expo: " << x << "^" << (1 << count) << endl;
    a = (a * a) % mod;
    b >>= 1;
    count++;
  }
  cout << "------------------------------------------------" << endl;
  return result;
}
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
int encryption(int e, int m, int n) {
  if (m < n) {
    int cipher = power(m, e, n);
    return cipher;
  } else {
    return -1;
  }
}
int gcd(int a, int b) {
  while (b != 0) {
    int temp = b;
    b = a % b;
    a = temp;
  }
  return a;
}
int decryption(int d, int c, int n) {
  int message = power(c, d, n);
  return message;
}

int extendedEuclidean(int r1, int r2) {
  int t1 = 0;
  int t2 = 1;
  while (r2 > 0) {
    int q = r1 / r2;
    int rem = r1 % r2;
    int t = t1 - (q * t2);
    r1 = r2;
    r2 = rem;
    t1 = t2;
    t2 = t;
  }
  return t1;
}

int calculateD(int e, int phi) {
  int d = extendedEuclidean(phi, e);
  if (d < 0) {
    d += phi;
  }
  return d;
}
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
int calculateE(int phi) {
  for (int i = 2; i < phi; i++) {
    if (gcd(phi, i) == 1) {
      return i;
    }
  }
  return -1;
}

bool isPrime(int num) {
  if (num <= 1) {
    return false;
  }
  for (int i = 2; i * i <= num; i++) {
    if (num % i == 0) {
      return false;
    }
  }
  return true;
}

int main() {
  cout << "Enter the value of p: ";
  int p;
  cin >> p;

  cout << "Enter the value of q: ";
  int q;
  cin >> q;

  if (isPrime(p) && isPrime(q)) {
    int n = p * q;
    int phi = (p - 1) * (q - 1);
    int e = calculateE(phi);
    int d = calculateD(e, phi);

    cout << "phi(n): " << phi << endl;
    cout << "e: " << e << endl;
    cout << "d: " << d << endl;

    int message;
    cout << "Enter the Message: ";
```

Roll No: 2103163
Batch: C32
Name: Om Shete

```
    cin >> message;

    int cipher = encryption(e, message, n);
    if (cipher != -1) {
      cout << "Encrypted Text: " << cipher << endl;
      int decryptedMessage = decryption(d, cipher, n);
      cout << "Decrypted Text: " << decryptedMessage << endl;
    } else {
      cout << "Encryption not possible, choose big p and q." << endl;
    }
  } else {
    cout << "Both p and q should be prime." << endl;
  }

  return 0;
}
```

Results:

```
Enter the value of p: 13
Enter the value of q: 17
phi(n): 192
e: 5
d: 77
Enter the Message: 8
-------------------------------------------------
Binary Expo: 8^1
Binary Expo: 8^4
-------------------------------------------------
Encrypted Text: 60
-------------------------------------------------
Binary Expo: 60^1
Binary Expo: 60^4
Binary Expo: 60^8
Binary Expo: 60^64
-------------------------------------------------
Decrypted Text: 8
```