2020BTECS00210

Cryptography and Network Security Lab

Batch – B4

Title:

Implementation of RSA Factorization

Theory:

- In mathematics, the Euclidean algorithm, or Euclid's algorithm, is an efficient method for computing the greatest common divisor (GCD) of two integers (numbers), the largest number that divides them both without a remainder.
- The Euclidean algorithm is based on the principle that the greatest common divisor of two numbers does not change if the larger number is replaced by its difference with the smaller number. For example, 21 is the GCD of 252 and 105 (as 252 = 21 × 12 and 105 = 21 × 5), and the same number 21 is also the GCD of 105 and 252 − 105 = 147.
- By using the extended Euclidean algorithm, the GCD can be expressed as a linear combination of the two original numbers, that is the sum of the two numbers, each multiplied by an integer (for example, 21 = 5 × 105 + (−2) × 252). The fact that the GCD can always be expressed in this way is known as Bézout's identity.
- The Euclidean algorithm has many theoretical and practical applications. It is used for reducing fractions to their simplest form and for performing division in modular arithmetic. Computations using this algorithm form part of the cryptographic protocols that are used to secure internet communications, and in methods for breaking these cryptosystems by factoring large composite numbers.
- In arithmetic and computer programming, the extended Euclidean algorithm is an extension to the Euclidean algorithm, and computes, in addition to the greatest common divisor (gcd) of integers a and b, also the coefficients of Bézout's identity, which are integers x and y such that

  $$ax + by = \gcd(a, b).$$

- The extended Euclidean algorithm is particularly useful when a and b are coprime. With that provision, x is the modular multiplicative inverse of a modulo b, and y is the modular multiplicative inverse of b modulo a.

CODE:

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef long long int ll;

// function to find gcd of two integer numbers
ll gcd(ll a, ll b)
{
    if (!a)
        return b;
    return gcd(b % a, a);
}

ll reduceB(ll a, char b[])
{
    // Initialize result
    ll mod = 0;

    // calculating mod of b with a to make
    // b like 0 <= b < a
    for (int i = 0; i < strlen(b); i++)
        mod = (mod * 10 + b[i] - '0') % a;

    return mod; // return modulo
}

ll gcdLarge(ll a, char b[])
{
    // Reduce 'b' (second number) after modulo with a
    ll num = reduceB(a, b);

    // gcd of two numbers
    return gcd(a, num);
}

int main()
{
    // first number which is integer
    ll a = 1221;

    char b[] = "12345678910111213141516171819202122232425262728 29";

    cout<<"Enter a Smaller Number: ";
    cin>>a;

    cout<<"Enter a Large Number: ";
    cin>>b;

    cout<<"\nThe GCD of Given Number is: ";
```

```
    if (a == 0)
        cout << b << endl;
    else
        cout << gcdLarge(a, b) << endl;

    return 0;
}
```

OUTPUT:

```
PS C:\Users\Admin\Desktop\WCE VII\CNS LAB\Assignment 13> cd "c:\Users\Admin\Desktop\WCE VII\CNS LAB\Assignment 13
\" ; if ($?) { g++ RSA_Factorization.cpp -o RSA_Factorization } ; if ($?) { .\RSA_Factorization }
Enter a Smaller Number: 741
Enter a Large Number: 96325841478

The GCD of Given Number is: 3
PS C:\Users\Admin\Desktop\WCE VII\CNS LAB\Assignment 13>
```