Cryptography and Network Security

Name: Piyush Mhaske Batch: B3
PRN : 2019BTECS00089

Assignment 12 : RSA Factorization challenge

Problem statement : RSA Algorithm

Theory :

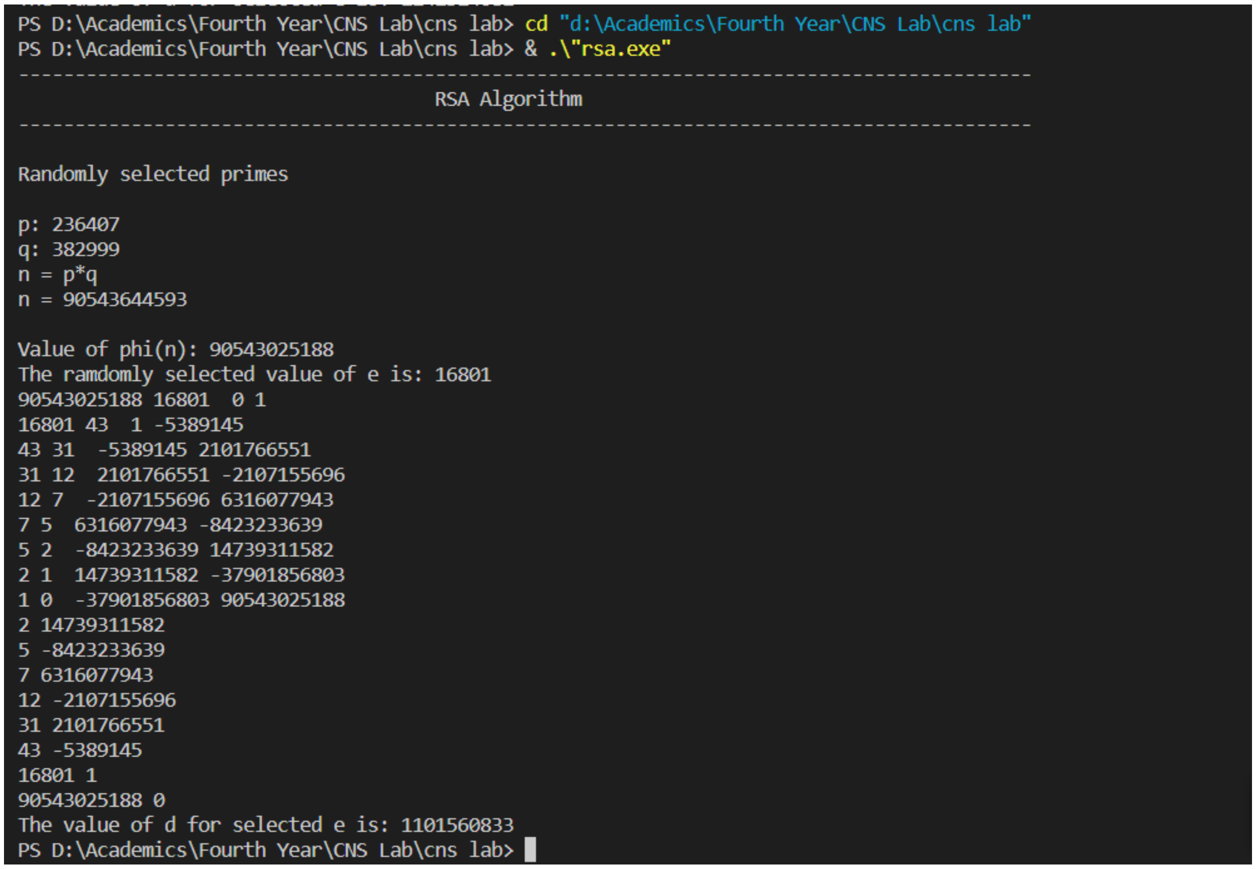
The idea of RSA is based on the fact that it is difficult to factorize a large integer. The public key consists of two numbers where one number is a multiplication of two large prime numbers. And private key is also derived from the same two prime numbers. So if somebody can factorize the large number, 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.

Code:

```
#include <bits/stdc++.h>
#define N 1000000
using namespace std;

long long power(long long a, long long b, long long mod){
    long long result = 1;
    while(b > 0){
        // check if the last bit is odd
        if(b&1)
            result = (result*a)%mod;
        a = (a*a)%mod;
        // b /= 2
        b >>= 1;
    }
    return result;
}
```

```
long long gcdExtended(long long a, long long b, long long *x, long long *y)
{
  cout << a << " " << b << " "
     << " " << *x << " " << *y << "\n";
  // Base Case
  if (b == 0)
  {
    return *x;
  }
  long long q = a / b;
  long long x1 = *y;
  long long y1 = *x - q * (*y);
  long long t1 = gcdExtended(b, a % b, &x1, &y1);

  cout << a << " " << *x << "\n";
  if (*x == 0 && t1 < 0)
    return a + t1;
  else
    return t1;

  // return gcd;
}

void SieveOfEratosthenes(int n, vector<int> &primes) {

    bool prime[n + 1];
    memset(prime, true, sizeof(prime));

    for(int p = 2; p * p <= n; p++) {
        if (prime[p]) {
            for (int i = p * p; i <= n; i += p)
                prime[i] = false;
        }
    }
    for (int p = 2; p <= n; p++)
        if (prime[p]){
            primes.push_back(p);
        }

}


int main() {

    char patternChar = '-';
    char resetChar = ' ';
    int lineWidth = 90;
    int initialWidth = 50;

    cout << setfill(patternChar) << setw(lineWidth) << patternChar << endl;
    cout << setfill(resetChar);
    cout << setw(initialWidth) << "RSA Algorithm" << endl;
    cout << setfill(patternChar) << setw(lineWidth) << patternChar << endl;
    cout << setfill(resetChar);
    vector<int> primes;
    // generating primes between 1 and N;
    SieveOfEratosthenes(N, primes);

    srand(time(0));
    // choose any two primes randomly
    int p, q;
    int primesSize = primes.size();
    int rand = std::rand();
    p = primes[(rand % primesSize)];
    do{
        rand = std::rand();
        q = primes[(rand % primesSize)];
    }while(p == q);

    cout << "\nRandomly selected primes\n" << endl;
    cout << "p: " << p << endl;
    cout << "q: " << q << endl;

    // calculate the value of n
    long long n = p*1LL*q;
    cout << "n = p*q" << endl;
    cout << "n = " << n << endl;


    // calculate the value of phi
    long long phi = (p-1)*1LL*(q-1);
    cout << "\nValue of phi(n): " << phi << endl;

    // generating all the co-primes between 2 and phi👎
    // acquire prime (a) such that a*a < phi value
    // store them

    vector<int> primeList;
    for(size_t i = 0; i < primes.size(); i++){
        if(primes[i]*1LL*primes[i] <= phi){
            primeList.push_back(primes[i]);
        }
    }

    // find the factors of unique prime factors of phu value
    vector<int> phiPrimeList;
    for(size_t i =0; i < primeList.size(); i++){
        if(phi > primeList[i] && (phi % primeList[i] == 0)){
            phiPrimeList.push_back(primeList[i]);
            while(phi % primeList[i] == 0){
                phi /= primeList[i];
            }
        }
    }

    if(phi > 1){
        phiPrimeList.push_back(phi);
    }

    // reassining the value of phi
    phi = (p-1)*1LL*(q-1);
    long long sizeRestriction  = 1e6;

    sizeRestriction = min(sizeRestriction, phi);

    // note : We are restricting the random coPrime upto 1e6
    vector<int> coPrimesOfPhi;

    vector<bool> phiVec(sizeRestriction, true);
    phiVec[0] = phiVec[1] = false;

    for(auto prime : phiPrimeList){
        for(int i = prime; i < sizeRestriction; i += prime){
            phiVec[i] = false;
        }
    }

    for(size_t i = 0; i < phiVec.size(); i++){
        if(phiVec[i])
            coPrimesOfPhi.push_back(i);
    }

    // cout << "Co-Primes between [2,maxLimit of restriction) are as follows: " << endl;

    // for(size_t i = 0; i < coPrimesOfPhi.size(); i++){
    //     cout << coPrimesOfPhi[i] << " ";
    // }
    // cout << endl;

    // avoiding selecting the first or any specific number of coprime which occured
    rand = std::rand();
    int e = coPrimesOfPhi[rand%coPrimesOfPhi.size()];
    cout << "The ramdomly selected value of e is: " << e << endl;

    long long x,y;
    x=0;
    y=1;
    int d = gcdExtended(phi, e, &x, &y);
    cout << "The value of d for selected e is: " << d << endl;

    return 0;
}
```

Ouput :



```
PS D:\Academics\Fourth Year\CNS Lab\cns lab> cd "d:\Academics\Fourth Year\CNS Lab\cns lab"
PS D:\Academics\Fourth Year\CNS Lab\cns lab> & .\"rsa.exe"
-----------------------------------------------------------------------------------------
                                     RSA Algorithm
-----------------------------------------------------------------------------------------

Randomly selected primes

p: 236407
q: 382999
n = p*q
n = 90543644593

Value of phi(n): 90543025188
The ramdomly selected value of e is: 16801
90543025188 16801  0 1
16801 43  1 -5389145
43 31  -5389145 2101766551
31 12  2101766551 -2107155696
12 7  -2107155696 6316077943
7 5  6316077943 -8423233639
5 2  -8423233639 14739311582
2 1  14739311582 -37901856803
1 0  -37901856803 90543025188
2 14739311582
5 -8423233639
7 6316077943
12 -2107155696
31 2101766551
43 -5389145
16801 1
90543025188 0
The value of d for selected e is: 1101560833
PS D:\Academics\Fourth Year\CNS Lab\cns lab> 
```