U18CO018
Shubham Shekhaliya
CNS
Assignment – 8

Write a program to implement the following RSA functions with large prime numbers.
(a) Key Generation
(b) Encryption
(c) Decryption
User can input plaintext as a number or text.

Code:-

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

#define int long long

mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());

int p, q, n, e, d, phi;
int upperBound = 1e9;

bool isPrime(int n) {
    if(n <= 1) {
        return false;
    }

    for(int i = 2;i <= sqrt(n);i++) {
        if(n % i == 0) {
            return false;
        }
    }

    return true;
}

int extended_euclidean(int a, int b, int& x, int& y) {
    if (b == 0) {
        x = 1;
        y = 0;
        return a;
    }
```

```
    int x1, y1;
    int d = extended_euclidean(b, a % b, x1, y1);
    x = y1;
    y = x1 - y1 * (a / b);
    return d;
}

int inverse(int a, int m) {
    int x, y;
    int g = extended_euclidean(a, m, x, y);
    return (x % m + m) % m;
}

__int128 power(__int128 x, __int128 n, __int128 mod) {
    if(n == 0) {
        return 1;
    }
    if(n % 2 == 0) {
        return power((x * x) % mod, n / 2, mod);
    }

    return (x * power((x * x) % mod, (n - 1) / 2, mod)) % mod;
}

void keyGeneration() {

    do {
        p = rng() % upperBound;
    }
    while(!isPrime(p));

    do {
        q = rng() % upperBound;
    }
    while(!isPrime(q));

    n = p * q;
    phi = (p - 1) * (q - 1);

    do {
        e = rng() % phi;
        if(!e) {
            e += 2;
        }
    }
```

```
    while(__gcd(e, phi) != 1);

    d = inverse(e, phi);

    cout << "Public Encryption key: { " << e << ", " << n << " }\n\n";
}

vector <int> encrypt(string message) {
    vector <int> cipher;
    for(char c: message) {
        cipher.push_back(power((c-65), e, n));
    }
    return cipher;
}

string decrypt(vector <int> cipher) {
    string message;

    for(int x: cipher) {
        message.push_back(char(power(x, d, n) + 65));
    }

    return message;
}

int32_t main() {

    while(true) {
        cout << "1)Key Generation\n2)Encryption\n3)Decryption\n4)Exit\n";

        int choice;
        cin >> choice;

        if(choice == 1) {
            keyGeneration();
        }
        else if(choice == 2) {
            cout << "Enter the message to encrpyt: ";
            string message;
            cin >> message;
            vector <int> cipher = encrypt(message);
            cout<< "Cipher is: ";
            for(int x: cipher) {
                cout << x << " ";
            }
```

```
            cout << "\n\n";
        }
        else if(choice == 3) {
            vector <int> cipher;
            cout << "Enter list of numbers to decrypt and -1 to stop: ";
            while(true) {
                int num;
                cin >> num;
                if(num == -1) {
                    break;
                }
                cipher.push_back(num);
            }
            string message = decrypt(cipher);
            cout << "Message is: " << message << "\n\n";
        }
        else if(choice == 4) {
            cout << "Thanks!\n\n";
            break;
        }
        else {
            cout << "Invalid choice!\n";
        }
    }

    return 0;
}
```

Output:-

```
PS D:\Course-Work\7th SEM\CNS\Assignment-8> cd "d:\Course-Work\7th SEM\CNS\Assignment-8\" ; if ($?) { g++ CNS8.cpp -o CNS8 } ; if
($?) { .\CNS8 }
1)Key Generation
2)Encryption
3)Decryption
4)Exit
1
Public Encryption key: { 174490404781434779, 480385657421013293 }

1)Key Generation
2)Encryption
3)Decryption
4)Exit
2
Enter the message to encrpyt: HELLO
Cipher is: 13163702329075401 354122596112385085 308187073167036548 308187073167036548 23771221592289995

1)Key Generation
2)Encryption
3)Decryption
4)Exit
3
Enter list of numbers to decrypt and -1 to stop: 13163702329075401 354122596112385085 308187073167036548 308187073167036548 237712
21592289995 -1
Message is: HELLO

1)Key Generation
2)Encryption
3)Decryption
4)Exit
4
Thanks!

PS D:\Course-Work\7th SEM\CNS\Assignment-8>
```