**Практическое занятие №6**

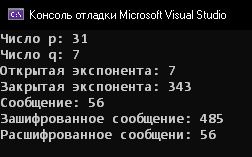
**Тема «Криптографическая защита информации»**

**Цель:** Овладение основными криптографическими алгоритмами асимметричного шифрования.

**Задание 1.**

Написать программу, которая зашифрует и расшифрует текст, введенный с клавиатуры, зашифрует и расшифрует его алгоритмом RSA.

Результат выполнения программы:



**Рисунок 6.1. Реализация алгоритма RSA**

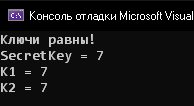
**Ключи использовались из строки 10.** [**Программа написана на языке C#.**](#_Реализация_алгоритма_RSA)

**Задание 2.**

Написать программу, которая получит общий секретный ключ из доступных публичных ключей с помощью алгоритма Диффи-Хеллмана.

Генерируйте секретный ключ Алисы и Боба, и высчитайте секретный ключ, удостоверьтесь что его одинаково смогут получить и Алиса и Боб.

Результат выполнения программы:



**Рисунок 6.2. Реализация алгоритма Диффи-Хеллмана**

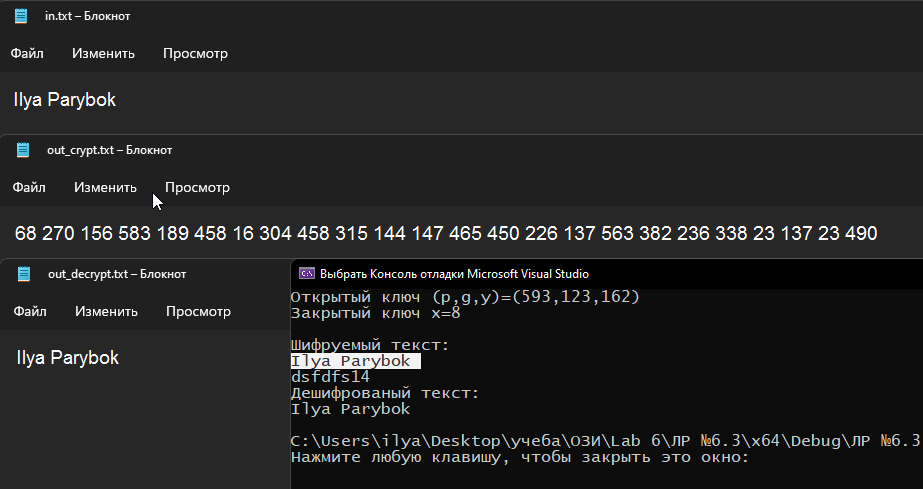
[Программа написаная на языке C#.](#_Реализация_алгоритма_Диффи-Хеллмана)

**Задание 3.**

Написать программу, которая сгенерирует ключи, зашифрует и расшифрует текст, введенный с клавиатуры алгоритмом Эль-Гамаля.

В приложении приведен пример программы реализующий данный алгоритм, напишите на его примере программу обрабатывающую нашу задачу.

Результат выполнения программы:



**Рисунок 6.2. Реализация алгоритма Эль-Гамаля**

[Программа написана на языке C++.](#_Реализация_алгоритма_Эль-Гамаля)

**Вывод:** в ходе лабораторной работы я изучил и программно реализовал следующие алгоритмы шифрования: Эль-Гамаля, RSA, Диффи-Хеллмана.

**Приложение**

# Реализация алгоритма Диффи-Хеллмана

class Program

{

static void Main(string[] args)

{

List<int> Keys = new List<int>();

Random rand = new Random();

for (int i = 0; i < 2; i++)

Keys.Add(rand.Next(1, 30));

for (int i = 0; i < 2; i++)

Keys.Add(rand.Next(1, 10));

//GetSecretKey(Keys[0], Keys[1], Keys[2], Keys[3]);

GetSecretKey(5, 23, 4, 3);

}

public static void GetSecretKey(int g, int p, int a, int b)

{

int A = Power(g, a, p);

int B = Power(g, b, p);

int K1 = Power(B, a, p);

int K2 = Power(A, b, p);

int SecretKey = Power(g, a \* b, p);

if (K1 == K2 && SecretKey == K1)

Console.WriteLine($"Ключи равны!\nSecretKey = {SecretKey}\nK1 = {K1}\nK2 = {K2}");

else

Console.WriteLine($"Ключи не равны!\nSecretKey = {SecretKey}\nK1 = {K1}\nK2 = {K2}");

Console.WriteLine(Power(2, 7, 37));

}

public static int Power(int a, int b, int n)

{

int tmp = a;

int sum = tmp;

for (int i = 1; i < b; i++)

{

for (int j = 1; j < a; j++)

{

sum += tmp;

if (sum >= n)

{

sum -= n;

}

}

tmp = sum;

}

return tmp;

} }

# Реализация алгоритма RSA

class Program

{

static void Main(string[] args)

{

Random rnd = new Random();

// Выбрать два простых различных числа

int p = 31;

int q = 7;

// Выбрать открытую экспоненту

int e = 47;

int msg = rnd.Next(5, 100);

// Вычислить модуль(произведение)

int n = p \* q;

// Вычислить функцию Эйлера

int EulerFunction = (p - 1) \* (q - 1);

//Вычислить секретную экспоненту

int d = modinv(e, EulerFunction);

e = 7;

n = 533;

d = 343;

List<int> OpenKey = new List<int>();

OpenKey.Add(e);

OpenKey.Add(n);

List<int> SecretKey = new List<int>();

SecretKey.Add(d);

SecretKey.Add(n);

Console.WriteLine("Число p: " + p);

Console.WriteLine("Число q: " + q);

Console.WriteLine("Открытая экспонента: " + e);

Console.WriteLine("Закрытая экспонента: " + d);

Console.WriteLine("Сообщение: " + msg);

Console.WriteLine("Зашифрованное сообщение: " + Encrypt(OpenKey, msg));

Console.WriteLine("Расшифрованное сообщени: " + Decrypt(SecretKey, Encrypt(OpenKey, msg)));

}

public static int? GetPrime(int start, int end)

{

List<int> Primes = new List<int>();

for (int i = start; i <= end; i++)

{

bool b = true;

for (int j = 2; j < i / 2; j++)

{

if (i % j == 0 & i % 1 == 0)

{

b = false;

}

}

if (b)

{

Primes.Add(i);

}

}

if (Primes.Count == 0)

return null;

else

{

Random rnd = new Random();

return Primes[rnd.Next(0, Primes.Count)];

}

}

public static int Encrypt(List<int> OpenKey, int msg)

{

return Modpow(msg, OpenKey[0], OpenKey[1]);

}

public static int Decrypt(List<int> SecretKey, int msg)

{

return Modpow(msg, SecretKey[0], SecretKey[1]);

}

public static int Modpow(int Base, int exp, int modulus)

{

Base %= modulus;

int result = 1;

while (exp > 0)

{

if ((exp & 1) != 0) result = (result \* Base) % modulus;

Base = (Base \* Base) % modulus;

exp >>= 1;

}

return result;

}

public static int modinv(int u, int v)

{

int inv, u1, u3, v1, v3, t1, t3, q;

int iter;

u1 = 1;

u3 = u;

v1 = 0;

v3 = v;

iter = 1;

while (v3 != 0)

{

q = u3 / v3;

t3 = u3 % v3;

t1 = u1 + q \* v1;

u1 = v1;

v1 = t1;

u3 = v3;

v3 = t3;

iter = -iter;

}

if (u3 != 1)

return 0;

if (iter < 0)

inv = v - u1;

else

inv = u1;

return inv;

}

}

# Реализация алгоритма Эль-Гамаля

using namespace std;

int power(int a, int b, int n) {// a^b mod n

int tmp = a;

int sum = tmp;

for (int i = 1; i < b; i++) {

for (int j = 1; j < a; j++) {

sum += tmp;

if (sum >= n) {

sum -= n;

}

}

tmp = sum;

}

return tmp;

}

int mul(int a, int b, int n) {// a\*b mod n

int sum = 0;

for (int i = 0; i < b; i++) {

sum += a;

if (sum >= n) {

sum -= n;

}

}

return sum;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

p - простое число

0 < g < p-1

0 < x < p-1

m - шифруемое сообщение m < p

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void crypt(int p, int g, int x, string inFileName, string outFileName) {

ifstream inf(inFileName.c\_str());

ofstream outf(outFileName.c\_str());

int y = power(g, x, p);

cout << "Открытый ключ (p,g,y)=" << "(" << p << "," << g << "," << y << ")" << endl;

cout << "Закрытый ключ x=" << x << endl;

cout << "\nШифруемый текст:" << endl;

while (inf.good()) {

int m = inf.get();

if (m > 0) {

cout << (char)m;

int k = rand() % (p - 2) + 1; // 1 < k < (p-1)

int a = power(g, k, p);

int b = mul(power(y, k, p), m, p);

outf << a << " " << b << " ";

}

}

cout << endl;

inf.close();

outf.close();

cout << "dsfdfs" << mul(power(32, 7, 37), 65, 37);

}

void decrypt(int p, int x, string inFileName, string outFileName) {

ifstream inf(inFileName.c\_str());

ofstream outf(outFileName.c\_str());

cout << "\nДешифрованый текст:" << endl;

while (inf.good()) {

int a = 0;

int b = 0;

inf >> a;

inf >> b;

if (a != 0 && b != 0) {

//wcout<<a<<" "<<b<<endl;

int deM = mul(b, power(a, p - 1 - x, p), p);// m=b\*(a^x)^(-1)mod p =b\*a^(p-1-x)mod p - трудно было найти нормальную формулу, в ней вся загвоздка

char m = static\_cast<char>(deM);

outf << m;

cout << m;

}

}

cout << endl;

inf.close();

outf.close();

}

int main()

{

setlocale(LC\_ALL, "RUSSIAN");

srand(time(NULL));

crypt(593, 123, 8, "in.txt", "out\_crypt.txt");

decrypt(593, 8, "out\_crypt.txt", "out\_decrypt.txt");

return 0;

}