Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

|  |
| --- |
| Институт космических и информационных технологий |
| *институт* |
| Кафедра Прикладной математики и компьютерной безопасности |
| *кафедра* |

**ЛАБОРАТОРНАЯ РАБОТА №3**

|  |
| --- |
| **Программная реализация метода ключевого обмена Диффи-Хеллмана** |
| *тема* |

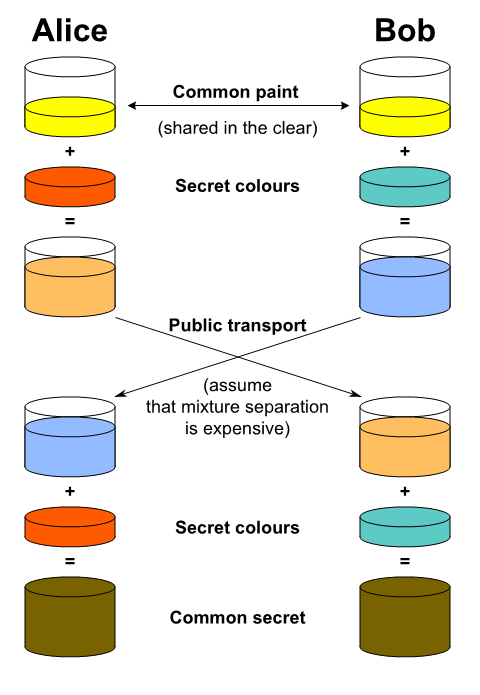
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Руководитель | | |  |  |  | В.И.Вайнштейн |
|  | |  |  | *подпись, дата* |  | *инициалы, фамилия* |
| Студент | КИ15-01 №031508683 | |  |  |  | М.С.Димаксян |
|  | *номер группы, зачетной книжки* | |  | *подпись, дата* |  | *инициалы, фамилия* |

Красноярск 2019

**Задание:**

Необходимо реализовать алгоритм ключевого обмена Диффи-Хеллмана.

**Теория:**

 Эта криптосистема была открыта в середине 70-х годов американскими учеными Диффи (Whitfield Diffie) и Хеллманом (Martin Hell-man) и привела к настоящей революции в криптографии и ее практических применениях. Это первая система, которая позволяла защищать информацию без использования секретных ключей, передаваемых по защищенным каналам.

Пусть строится система связи для абонентов ,,,… У каждого абонента есть своя секретная и открытая информация. Для организации этой системы выбирается большое простое число , где – простое, и некоторое число , , такое, что все числа из множества {} могут быть представлены как различные степени (известны различные подходы для нахождения таких чисел один из них будет представлен ниже). Числа и известны всем абонентам.

При произвольно заданном задача выбора может оказаться очень трудной, связанной с разложением на простые множители числа . Дело в том, что для обеспечения высокой стойкости рассмотренной системы число должно обязательно содержать большой простой множитель. Поэтому часто рекомендуют использовать следующий подход. Простое число р выбирается таким, чтобы выполнялось равенство , где — также простое число. Тогда в качестве можно взять любое число, для которого справедливы неравенства и .

Абоненты выбирают большие числа , которые хранят в секрете (обычно такой выбор рекомендуется проводить случайно, используя датчики случайных чисел). Каждый абонент вычисляет соответствующее число , которое открыто передается другим абонентам,

,

,

.

Допустим, абонент А решил организовать сеанс связи с , при этом обоим абонентам доступны открытые ключи. Абонент А сообщает B по открытому каналу, что он хочет передать ему сообщение. Затем абонент А вычисляет величину

(никто другой кроме А этого сделать не может, так как число ХА секретно). В свою очередь, абонент В вычисляет число

.

**Исходный код:**

**Функции теории чисел:**

public static class NumberTheory

{

public static BigInteger BinaryModPow(BigInteger Number, BigInteger Deg, BigInteger Mod)

{

BigInteger Result = 1;

BigInteger Bit = Number;

if (Bit < 0) Bit = (Number + Mod) % Mod;

while (Deg > 0)

{

if ((Deg & 1) == 1)

{

Result \*= Bit;

Result %= Mod;

}

Deg >>= 1;

if (Deg > 0)

{

Bit \*= Bit;

Bit %= Mod;

}

}

return Result;

}

public static BigInteger GCD(BigInteger a, BigInteger b, out BigInteger x, out BigInteger y)

{

if (a == 0)

{

x = 0;

y = 1;

return b;

}

BigInteger x1, y1;

BigInteger d = GCD(b % a, a, out x1, out y1);

x = y1 - (b / a) \* x1;

y = x1;

if (d < 0)

{

d = -d;

x = -x;

y = -y;

}

return d;

}

public static BigInteger Foo(BigInteger element, BigInteger Mod)

{

if (element < 0) element = (element + Mod) % Mod;

BigInteger x, y;

BigInteger g = GCD(element, Mod, out x, out y);

//Обратный элемент не существует, если кольцо не является полем (модуль не является степенью простого числа) и элемент не взаимно прост с модулем

//Возвращает 0 в случае, если обратный элемент не существует

if (g != 1) x = 0;

//Приведем результат в положительный вид по модулю

x = x % Mod;

if (x < 0) x = (x + Mod) % Mod;

return x;

}

}

**Тест Миллера-Рабина:**

public static class RabinMiller

{

// Random generator (thread safe)

private static ThreadLocal<Random> s\_Gen = new ThreadLocal<Random>(

() =>

{

return new Random();

}

);

// Random generator (thread safe)

private static Random Gen

{

get

{

return s\_Gen.Value;

}

}

public static Boolean IsPrime(BigInteger value, int witnesses)

{

if (value < 5)

return (value == 2 || value == 3);

if (witnesses <= 0)

witnesses = 1;

BigInteger d = value - 1;

int s = 0;

while (d % 2 == 0)

{

d /= 2;

s += 1;

}

Byte[] bytes = new Byte[value.ToByteArray().LongLength];

BigInteger a;

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

{

do

{

Gen.NextBytes(bytes);

a = new BigInteger(bytes);

}

while (a < 2 || a >= value - 2);

BigInteger x = BigInteger.ModPow(a, d, value);

if (x == 1 || x == value - 1)

continue;

for (int r = 1; r < s; r++)

{

x = BigInteger.ModPow(x, 2, value);

if (x == 1)

return false;

if (x == value - 1)

break;

}

if (x != value - 1)

return false;

}

return true;

}

}

**Параметры метода Диффи-Хеллмана:**

public class DiffieHellman

{

public struct Params

{

static public BigInteger P { get; set; }

static public BigInteger Q { get; set; }

static public BigInteger G { get; set; }

}

}

**Функция интерфейса:**

private void f3\_textParam\_KeyPress(object sender, KeyPressEventArgs e)

{

if (MyAlphabets.Numeric.IndexOf(e.KeyChar) == -1 && e.KeyChar != (char)8) e.Handled = true;

}

private void f3\_textQ\_Validated(object sender, EventArgs e)

{

if (f3\_textQ.TextLength > 0)

{

BigInteger Q = BigInteger.Parse(f3\_textQ.Text);

if (Q % 2 == 0) { Q++; f3\_textQ.Text = Convert.ToString(Q); }

f3\_textP.Text = Convert.ToString(Q \* 2 + 1);

}

}

private void f3\_textP\_TextChanged(object sender, EventArgs e)

{

f3\_textG.Text = "";

}

private void f3\_textSecretA\_TextChanged(object sender, EventArgs e)

{

f3\_textOpenA.Text = "";

f3\_textOpenA\_inWindowB.Text = "";

f3\_textCommonSecret\_inWindowA.Text = "";

f3\_textCommonSecret\_inWindowB.Text = "";

}

private void f3\_textSecretB\_TextChanged(object sender, EventArgs e)

{

f3\_textOpenB.Text = "";

f3\_textOpenB\_inWindowA.Text = "";

f3\_textCommonSecret\_inWindowA.Text = "";

f3\_textCommonSecret\_inWindowB.Text = "";

}

**Функция поиска следующего простого q:**

private void f3\_buttonNextPrime\_Click(object sender, EventArgs e)

{

f3\_textQ.Text = new string(f3\_textQ.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger ParamP, ParamQ;

if (f3\_textQ.TextLength > 0)

ParamQ = BigInteger.Parse(f3\_textQ.Text);

else

ParamQ = 0;

if (++ParamQ % 2 == 0) ParamQ++;

while (!RabinMiller.IsPrime(ParamQ, 3) || !RabinMiller.IsPrime(ParamP = 2 \* ParamQ + 1, 3))

ParamQ += 2;

f3\_textP.Text = Convert.ToString(ParamP);

f3\_textQ.Text = Convert.ToString(ParamQ);

}

**Функция поиска g:**

private void f3\_buttonNextGenerator\_Click(object sender, EventArgs e)

{

f3\_textP.Text = new string(f3\_textP.Text.Where(t => char.IsDigit(t)).ToArray());

f3\_textQ.Text = new string(f3\_textQ.Text.Where(t => char.IsDigit(t)).ToArray());

f3\_textG.Text = new string(f3\_textG.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger ParamP, ParamQ, ParamG = 0;

if (f3\_textP.TextLength > 0 && f3\_textQ.TextLength > 0 && RabinMiller.IsPrime(ParamP = BigInteger.Parse(f3\_textP.Text), 3)

&& RabinMiller.IsPrime(ParamQ = BigInteger.Parse(f3\_textQ.Text), 3))

{

RandomNumberGenerator rng = RandomNumberGenerator.Create();

while (ParamG < 2 || ParamG >= ParamP - 1 || NumberTheory.BinaryModPow(ParamG, ParamQ, ParamP) == 1)

{

ParamG = BigIntegerRandomUtils.RandomInRange(rng, 2, ParamP - 1);

}

f3\_textG.Text = Convert.ToString(ParamG);

}

}

**Установка параметров системы:**

private void f3\_buttonSetKeys\_Click(object sender, EventArgs e)

{

f3\_textP.Text = new string(f3\_textP.Text.Where(t => char.IsDigit(t)).ToArray());

f3\_textQ.Text = new string(f3\_textQ.Text.Where(t => char.IsDigit(t)).ToArray());

f3\_textG.Text = new string(f3\_textG.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger p = 0, q = 0, g = 0;

if (f3\_textP.TextLength > 0 && f3\_textQ.TextLength > 0 && f3\_textG.TextLength > 0 &&

RabinMiller.IsPrime(p = BigInteger.Parse(f3\_textP.Text), 10) &&

RabinMiller.IsPrime(q = BigInteger.Parse(f3\_textQ.Text), 10) &&

((g = BigInteger.Parse(f3\_textG.Text)) >= 2) && g < p - 1 && NumberTheory.BinaryModPow(g, q, p) != 1)

{

DiffieHellman.Params.P = p;

DiffieHellman.Params.Q = q;

DiffieHellman.Params.G = g;

f3\_buttonSetKeys.Enabled = false;

f3\_buttonClearKeys.Enabled = true;

f3\_buttonNextPrime.Enabled = false;

f3\_buttonNextGenerator.Enabled = false;

f3\_textQ.ReadOnly = true;

f3\_textG.ReadOnly = true;

f3\_button\_SetSecretA.Enabled = true;

f3\_button\_GetRandomSecretA.Enabled = true;

f3\_textSecretA.ReadOnly = false;

f3\_button\_SetSecretB.Enabled = true;

f3\_button\_GetRandomSecretB.Enabled = true;

f3\_textSecretB.ReadOnly = false;

}

}

**Сброс параметров системы:**

private void f3\_buttonClearKeys\_Click(object sender, EventArgs e)

{

f3\_buttonSetKeys.Enabled = true;

f3\_buttonClearKeys.Enabled = false;

f3\_buttonNextPrime.Enabled = true;

f3\_buttonNextGenerator.Enabled = true;

f3\_textQ.ReadOnly = false;

f3\_textG.ReadOnly = false;

f3\_button\_SetSecretA.Enabled = false;

f3\_button\_GetRandomSecretA.Enabled = false;

f3\_textSecretA.ReadOnly = true;

f3\_button\_SetSecretB.Enabled = false;

f3\_button\_GetRandomSecretB.Enabled = false;

f3\_textSecretB.ReadOnly = true;

f3\_textSecretA.Text = "";

f3\_textOpenA.Text = "";

f3\_textOpenB\_inWindowA.Text = "";

f3\_textCommonSecret\_inWindowA.Text = "";

f3\_textSecretB.Text = "";

f3\_textOpenB.Text = "";

f3\_textOpenA\_inWindowB.Text = "";

f3\_textCommonSecret\_inWindowB.Text = "";

}

**Генерация секретного ключа на примере абонента А:**

private void f3\_button\_GetRandomSecretA\_Click(object sender, EventArgs e)

{

RandomNumberGenerator rng = RandomNumberGenerator.Create();

f3\_textSecretA.Text = Convert.ToString(BigIntegerRandomUtils.RandomInRange(rng, DiffieHellman.Params.Q, DiffieHellman.Params.P));

}

**Установка введенного секретного ключа на примере абонента А и расчет общего секретного ключа в случае известного открытого ключа абонента Б:**

private void f3\_button\_SetSecretA\_Click(object sender, EventArgs e)

{

f3\_textSecretA.Text = new string(f3\_textSecretA.Text.Where(t => char.IsDigit(t)).ToArray());

if (f3\_textSecretA.TextLength > 0)

{

BigInteger Xa = BigInteger.Parse(f3\_textSecretA.Text);

f3\_textOpenA.Text = Convert.ToString(NumberTheory.BinaryModPow(DiffieHellman.Params.G,Xa, DiffieHellman.Params.P));

f3\_textOpenA\_inWindowB.Text = f3\_textOpenA.Text;

if (f3\_textOpenB\_inWindowA.TextLength > 0)

{

f3\_textCommonSecret\_inWindowA.Text =

Convert.ToString(NumberTheory.BinaryModPow(BigInteger.Parse(f3\_textOpenB\_inWindowA.Text), Xa, DiffieHellman.Params.P));

f3\_textCommonSecret\_inWindowB.Text = f3\_textCommonSecret\_inWindowA.Text;

}

}

}

**Результат работы программы:**

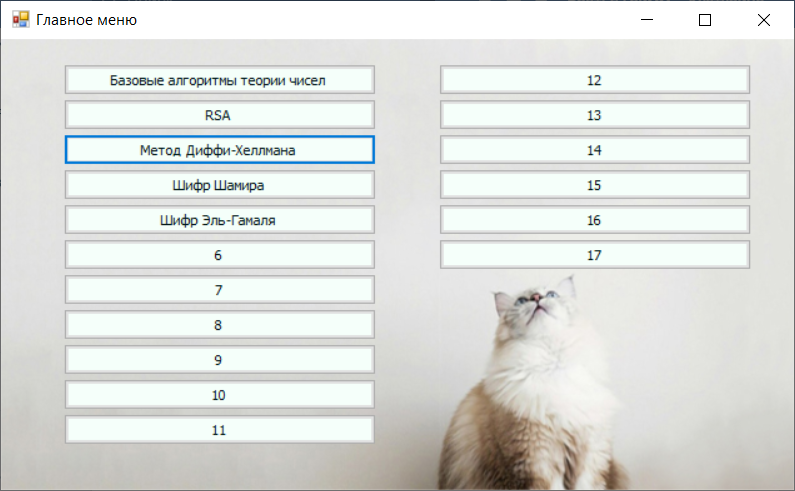


Рисунок 1. Главное меню программы.

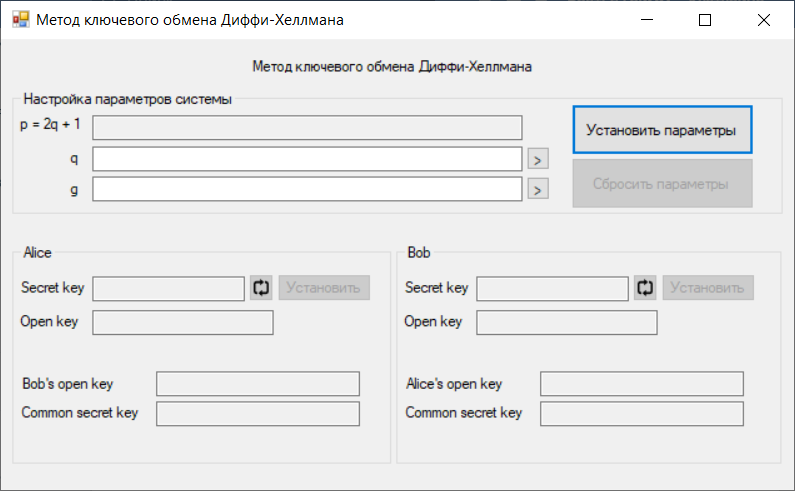


Рисунок 2. Окно метода Диффи-Хеллмана.

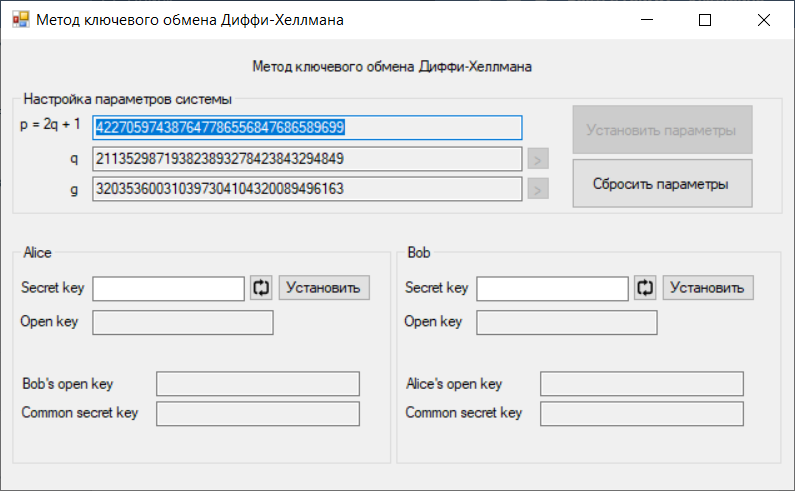


Рисунок 3. Установка параметров.

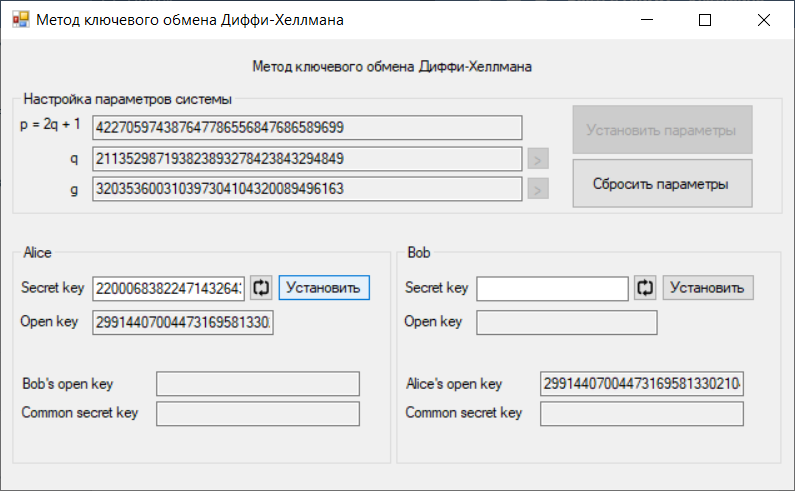


Рисунок 4. Установка секретного ключа Алисы.

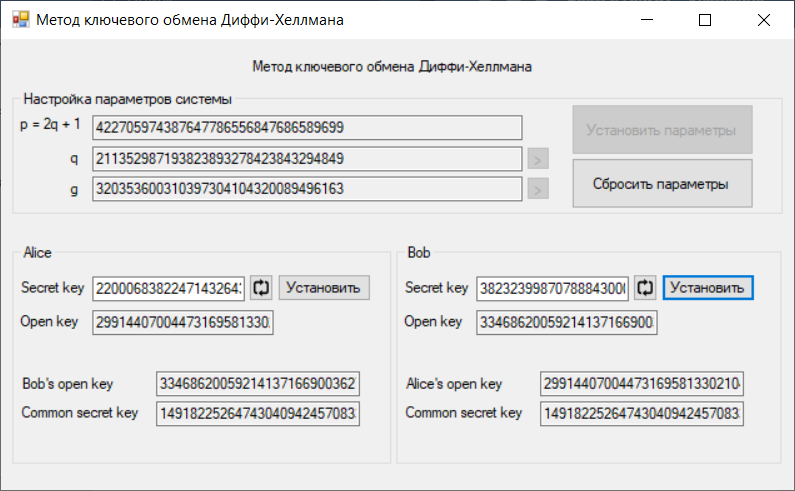


Рисунок 5. Установка секретного ключа Боба.

**Вывод:**

В ходе выполнения лабораторной работы я ознакомилась с алгоритмом работы метода ключевого обмена Диффи-Хеллмана, реализовала на практике программу обмену ключами на языке C#.