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 №2**

|  |
| --- |
| **Криптосистема RSA** |
| *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:**

Написать и отладить программу, реализующую алгоритм шифрования RSA для передачи секретных сообщений в адрес абонента B.

**Теория:**

Алгоритм шифрования RSA является алгоритмом с открытым ключом. Для генерации двух ключей (личного и открытого ) абоненту B необходимо выполнить следующие действия.

1. Выбрать два больших случайных простых числа и .
2. Вычислить и .
3. Выбрать случайным образом простое число , меньшее, чем , у которого нет общих делителей (кроме 1) с числом (взаимно простые числа). Числа и составляют открытый ключ абонента B:
4. С помощью обобщенного алгоритма Евклида вычислить число (инверсия по модулю ), такое что остаток от деления на был равен 1:

Числа и составляют личный ключ абонента

Абонент A шифрует сообщение по формуле

и пересылает шифротекст c участнику B по открытой линии.

Абонент B, получивший зашифрованное сообщение, вычисляет открытый текст по формуле

Для шифрования большого сообщения оно разбивается на маленькие блоки .

**Исходный код:**

**Функции интерфейса:**

private void f2\_numericP\_ValueChanged(object sender, EventArgs e)

{

f2\_numericP.ForeColor = Color.Black;

f2\_buttonGetKeys.Enabled = false;

f2\_buttonEncrypt.Enabled = false;

f2\_buttonDecrypt.Enabled = false;

f2\_textBoxAlf.ReadOnly = false;

f2\_textBoxKeyE.Text = "";

f2\_textBoxKeyD.Text = "";

f2\_textBoxKeyN.Text = "";

f2\_textBoxKeyN2.Text = "";

}

private void f2\_numericQ\_ValueChanged(object sender, EventArgs e)

{

f2\_numericQ.ForeColor = Color.Black;

f2\_buttonGetKeys.Enabled = false;

f2\_buttonEncrypt.Enabled = false;

f2\_buttonDecrypt.Enabled = false;

f2\_textBoxAlf.ReadOnly = false;

f2\_textBoxKeyE.Text = "";

f2\_textBoxKeyD.Text = "";

f2\_textBoxKeyN.Text = "";

f2\_textBoxKeyN2.Text = "";

}

private void f2\_buttonCheckPrimes\_Click(object sender, EventArgs e)

{

//проверка введенных пользователем чисел на простоту

if (IsPrime(Convert.ToDouble(f2\_numericP.Value)))

{

f2\_numericP.ForeColor = Color.Green;

f2\_buttonGetKeys.Enabled = true;

}

else

f2\_numericP.ForeColor = Color.Red;

if (IsPrime(Convert.ToDouble(f2\_numericQ.Value)))

f2\_numericQ.ForeColor = Color.Green;

else

{

f2\_numericQ.ForeColor = Color.Red;

f2\_buttonGetKeys.Enabled = false;

}

}

private void f2\_buttonGenPrimes\_Click(object sender, EventArgs e)

{

//выбор случайных простых из базы

f2\_numericP.Value = Convert.ToDecimal(pqPrimes[rnd.Next(pqPrimes.Count)]);

f2\_numericQ.Value = Convert.ToDecimal(pqPrimes[rnd.Next(pqPrimes.Count)]);

f2\_numericP.ForeColor = Color.Green;

f2\_numericQ.ForeColor = Color.Green;

f2\_buttonGetKeys.Enabled = true;

}

**Простая функция проверки на простоту:**

public static bool IsPrime(double number)

{

if (number <= 1) return false;

if (number == 2) return true;

if (number % 2 == 0) return false;

var boundary = (int)Math.Floor(Math.Sqrt(number));

for (int i = 3; i <= boundary; i += 2)

if (number % i == 0)

return false;

return true;

}

**Функция нахождения обратного элемента в кольце или поле:**

public static Decimal Foo(Decimal element, Decimal Mod)

{

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

Decimal x, y;

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

if (g == -1)

{

g = 1;

x = -x;

y = -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;

}

**Функция нахождения НОД:**

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

{

if (a == 0)

{

x = 0;

y = 1;

return b;

}

Decimal x1, y1;

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

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

y = x1;

return d;

}

**Функция генерации ключей:**

private void f2\_buttonGetKeys\_Click(object sender, EventArgs e)

{

Decimal P = f2\_numericP.Value;

Decimal Q = f2\_numericQ.Value;

Decimal N = P \* Q;

Decimal Z = (P - 1) \* (Q - 1); //Ф(N)

Decimal E = N - 1;

//BigInteger bigIntFromDouble = new BigInteger(179032.6541);

while (E >= N || !IsPrime(Convert.ToDouble(E)) || (GCD(E,Z,out Decimal x, out Decimal y) != 1))

{

E--;

if (E>10000000) E = Convert.ToDecimal(ePrimes[rnd.Next(ePrimes.Count)]);

}

f2\_textBoxKeyE.Text = Convert.ToString(E);

f2\_textBoxKeyN.Text = Convert.ToString(N);

Decimal D = Foo(E, Z);

f2\_textBoxKeyD.Text = Convert.ToString(D);

f2\_textBoxKeyN2.Text = Convert.ToString(N);

f2\_buttonEncrypt.Enabled = true;

f2\_buttonDecrypt.Enabled = true;

f2\_textBoxAlf.ReadOnly = true;

}

**Функция зашифрования:**

private void f2\_buttonEncrypt\_Click(object sender, EventArgs e)

{

string Source = f2\_Original.Text;

//Для удаления запрещенных символов составим строку из символов текста и ключа, отсутствующих в алфавите

List<char> prohibited\_symbs = new List<char>();

for (int i = Source.Length - 1; i >= 0; i--)

if (f2\_textBoxAlf.Text.IndexOf(Source[i]) < 0)

prohibited\_symbs.Add(Source[i]);

//Удалим все встретившиеся запрещенные символы

for (int i = 0; i < prohibited\_symbs.Count; i++)

{

Source = Source.Replace(Convert.ToString(prohibited\_symbs[i]), "");

}

f2\_Original.Text = Source;

//Шифрование

f2\_Ciphertext.Text = "";

for (int i = 0; i < Source.Length; i++)

{

f2\_Ciphertext.Text += BinaryModPow(f2\_textBoxAlf.Text.IndexOf(Source[i]), Convert.ToInt64(f2\_textBoxKeyE.Text), Convert.ToDecimal(f2\_textBoxKeyN.Text));

f2\_Ciphertext.Text += " ";

}

}

**Функция расшифрования:**

private void f2\_buttonDecrypt\_Click(object sender, EventArgs e)

{

List<string> codes = new List<string> (f2\_Ciphertext.Text.Split(' '));

f2\_Original.Text = "";

for (int i =0; i< codes.Count - 1;i++)

{

f2\_Original.Text += f2\_textBoxAlf.Text[Convert.ToInt32(BinaryModPow(Convert.ToDecimal(codes[i]), Convert.ToInt64(f2\_textBoxKeyD.Text), Convert.ToDecimal(f2\_textBoxKeyN.Text)))];

}

}

**Результат работы программы:**

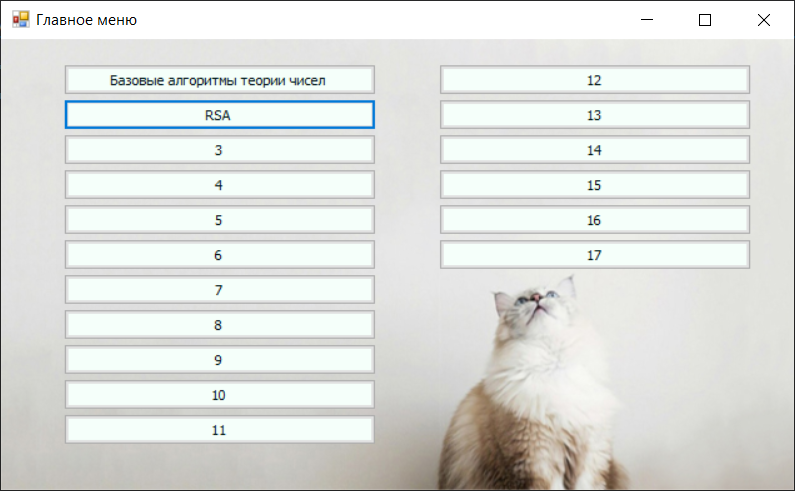


Рисунок 1. Главное меню программы.

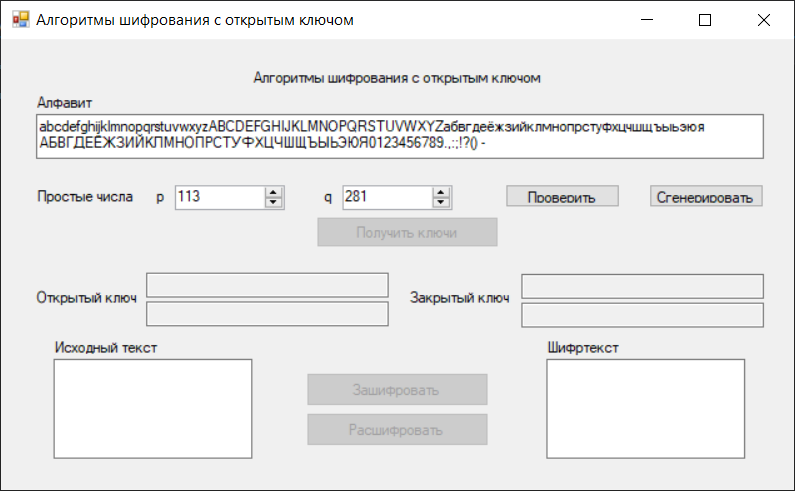


Рисунок 2. Окно RSA.

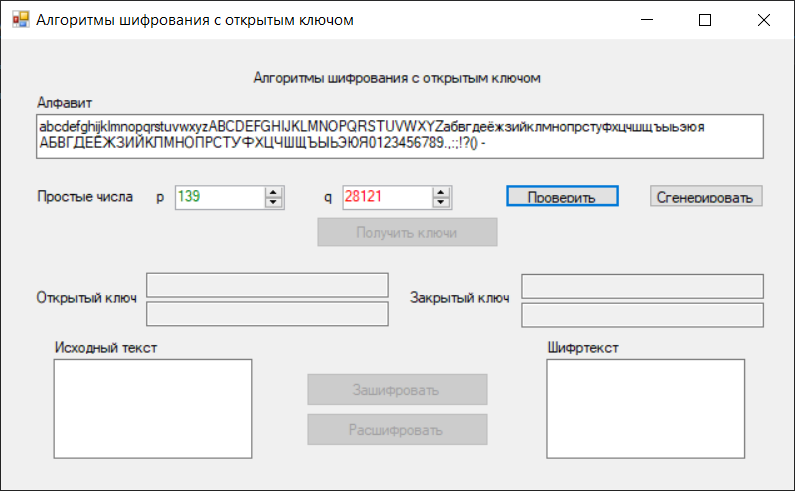


Рисунок 3. Проверка введенных чисел.

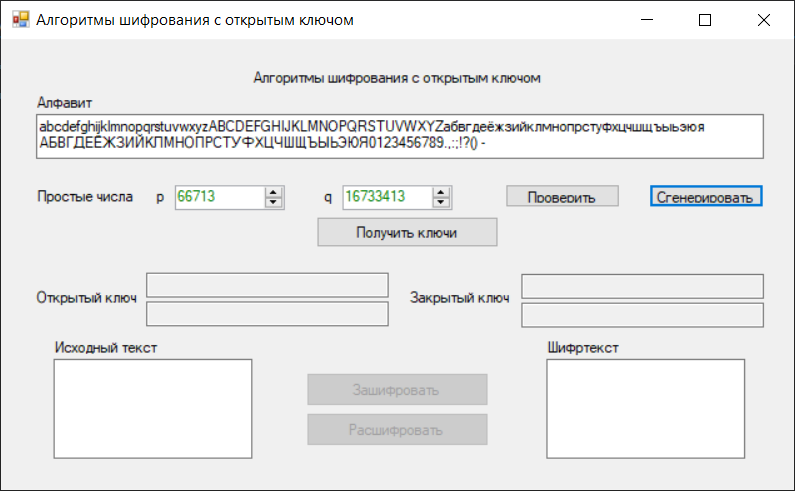


Рисунок 4. Генерация чисел.

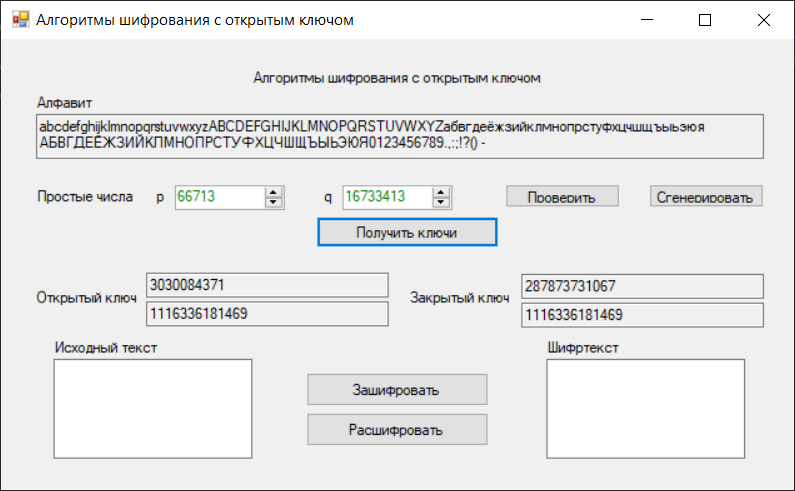


Рисунок 5. Получение ключей.

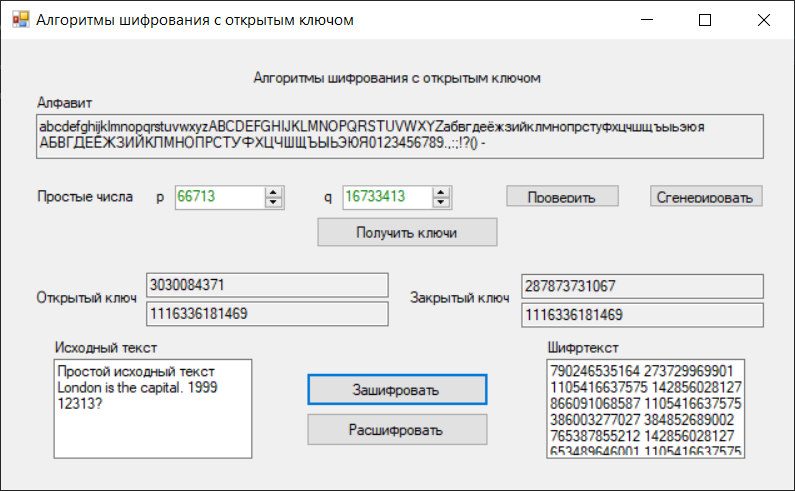


Рисунок 6. Зашифрование

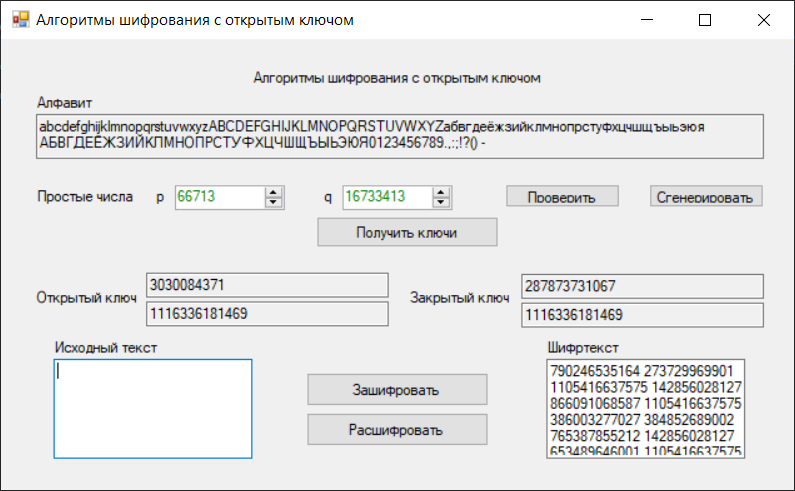


Рисунок 7. Очистка окна

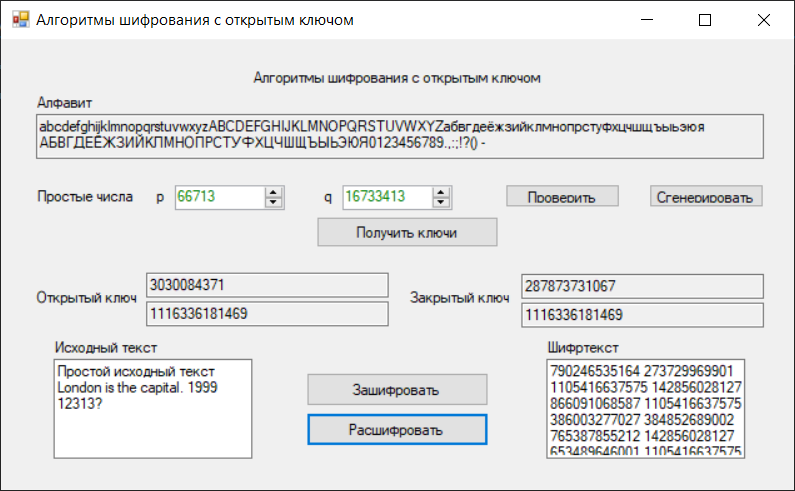


Рисунок 8. Расшифрование

**Вывод:**

В ходе выполнения лабораторной работы я ознакомилась с алгоритмом работы RSA, реализовала на практике программу зашифрованию и расшифрованию на языке C#.