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 №10**

|  |
| --- |
| Алгоритм электронной подписи на базе шифра Эль-Гамаля |
| *тема* |

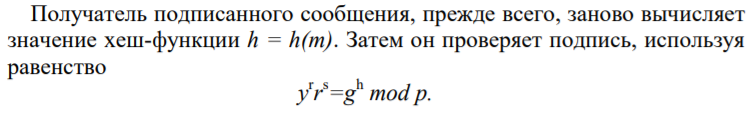
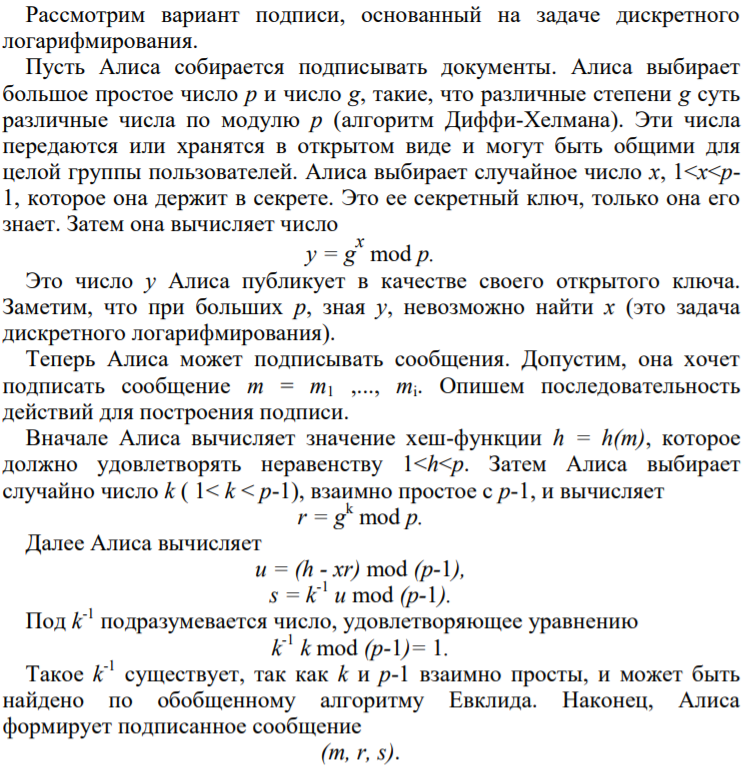
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Руководитель | | |  |  |  | В.И.Вайнштейн |
|  | |  |  | *подпись, дата* |  | *инициалы, фамилия* |
| Студент | КИ15-01 №031508683 | |  |  |  | М.С.Димаксян |
|  | *номер группы, зачетной книжки* | |  | *подпись, дата* |  | *инициалы,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

Красноярск 2019

**Задание:**

Изучить основные принципы построения и реализации алгоритма электронной подписи на базе шифра Эль-Гамаля.

**Теория:**

****

**Исходный код:**

**Функция подписи:**

private void f10\_buttonSign\_Click(object sender, EventArgs e)

{

BigInteger P = BigInteger.Parse(f10\_textP.Text);

BigInteger G = BigInteger.Parse(f10\_textG.Text);

BigInteger X = BigInteger.Parse(f10\_textSecretKey.Text);

BigInteger K = BigInteger.Parse(f10\_textSessionKey.Text);

//Открываем документ и вычисляем его хэш-функцию в переменную fileHash

string fileHash = "";

OpenFileDialog Load = new OpenFileDialog();

Load.InitialDirectory = "D:\\Учеба\\Крипта 2\\Labs\\Labs\\bin\\Debug\\TextFiles";

Load.DefaultExt = "txt";

Load.FileName = "in.txt";

Load.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

Load.FilterIndex = 2;

Load.RestoreDirectory = true;

Load.Multiselect = false;

string sFileName;

if (Load.ShowDialog() == DialogResult.OK)

{

sFileName = Load.FileName;

byte[] file = File.ReadAllBytes(Load.FileName);

//Вычисление хэш-строки

if (f10\_chooseHashFun.Text == "MD5")

fileHash = HashFunctions.MD5.GetHashMD5(file);

if (f10\_chooseHashFun.Text == "SHA-1")

fileHash = HashFunctions.SHA1.GetHashSHA1(file);

if (f10\_chooseHashFun.Text == "ГОСТ Р 34.11-2012 256 бит")

fileHash = HashFunctions.GOST.GetHashGOST(file, 256);

if (f10\_chooseHashFun.Text == "ГОСТ Р 34.11-2012 512 бит")

fileHash = HashFunctions.GOST.GetHashGOST(file, 512);

}

BigInteger hashDigit = new BigInteger(Encoding.Default.GetBytes(fileHash));

BigInteger r = NumberTheory.BinaryModPow(G, K, P);

BigInteger u = (hashDigit - X \* r) % (P - 1);

u = (u + (P - 1)) % (P - 1);

BigInteger antiK = NumberTheory.Foo(K, P - 1);

BigInteger s = (antiK \* u) % (P - 1);

if (s > 0 && r > 0)

{

if (hashDigit > P) MessageBox.Show(("Подпись небезопасна: используйте значение P, большее h(m) = " + Convert.ToString(hashDigit)), "Предупреждение");

f10\_textBoxSign.Text = Convert.ToString(s);

f10\_textBoxSignR.Text = Convert.ToString(r);

}

else

{

f10\_textBoxSign.Text = "";

f10\_textBoxSignR.Text = "";

}

}

#### Функция проверки подписи:

private void f10\_buttonCheckSign\_Click(object sender, EventArgs e)

{

//Удаление недопустимых символов

f10\_textOpenKeyRecieved.Text = new string(f10\_textOpenKeyRecieved.Text.Where(t => char.IsDigit(t)).ToArray());

f10\_textBoxSignS\_Recieved.Text = new string(f10\_textBoxSignS\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

f10\_textBoxSignR\_Recieved.Text = new string(f10\_textBoxSignR\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

f10\_textP\_Recieved.Text = new string(f10\_textP\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

f10\_textG\_Recieved.Text = new string(f10\_textG\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

//Считывание полученных параметров подписи

BigInteger OpenKey = 0;

if (f10\_textOpenKeyRecieved.TextLength > 0)

OpenKey = BigInteger.Parse(f10\_textOpenKeyRecieved.Text);

BigInteger SignS = 0;

if (f10\_textBoxSignS\_Recieved.TextLength > 0)

SignS = BigInteger.Parse(f10\_textBoxSignS\_Recieved.Text);

BigInteger SignR = 0;

if (f10\_textBoxSignR\_Recieved.TextLength > 0)

SignR = BigInteger.Parse(f10\_textBoxSignR\_Recieved.Text);

BigInteger P = 0;

if (f10\_textP\_Recieved.TextLength > 0)

P = BigInteger.Parse(f10\_textP\_Recieved.Text);

BigInteger G = 0;

if (f10\_textG\_Recieved.TextLength > 0)

G = BigInteger.Parse(f10\_textG\_Recieved.Text);

//Открываем документ и вычисляем его хэш-функцию в переменную fileHash

string fileHash = "";

OpenFileDialog Load = new OpenFileDialog();

Load.InitialDirectory = "D:\\Учеба\\Крипта 2\\Labs\\Labs\\bin\\Debug\\TextFiles";

Load.DefaultExt = "txt";

Load.FileName = "in.txt";

Load.Filter = "txt files (\*.txt)|\*.txt|All files (\*.\*)|\*.\*";

Load.FilterIndex = 2;

Load.RestoreDirectory = true;

Load.Multiselect = false;

string sFileName;

if (Load.ShowDialog() == DialogResult.OK)

{

sFileName = Load.FileName;

byte[] file = File.ReadAllBytes(Load.FileName);

//Вычисление хэш-строки

if (f10\_chooseHashFunRecieved.Text == "MD5")

fileHash = HashFunctions.MD5.GetHashMD5(file);

if (f10\_chooseHashFunRecieved.Text == "SHA-1")

fileHash = HashFunctions.SHA1.GetHashSHA1(file);

if (f10\_chooseHashFunRecieved.Text == "ГОСТ Р 34.11-2012 256 бит")

fileHash = HashFunctions.GOST.GetHashGOST(file, 256);

if (f10\_chooseHashFunRecieved.Text == "ГОСТ Р 34.11-2012 512 бит")

fileHash = HashFunctions.GOST.GetHashGOST(file, 512);

}

BigInteger hashDigit = new BigInteger(Encoding.Default.GetBytes(fileHash)); //Вычисляем хэш файла

if (OpenKey > 0 && SignS > 0 && SignR > 0)

{

BigInteger left = NumberTheory.BinaryModPow(OpenKey, SignR, P) \*

NumberTheory.BinaryModPow(SignR, SignS, P);

BigInteger right = NumberTheory.BinaryModPow(G, hashDigit, P);

if (left % P == right)

f10\_textVerificationResult.Text = "Подпись верна";

else

f10\_textVerificationResult.Text = "Подпись неверна";

}

else

{

f10\_textVerificationResult.Text = "Подпись неверна";

}

}

#### Функции установки параметров

private void f10\_buttonSetParams\_Click(object sender, EventArgs e)

{

f10\_textQ.Text = new string(f10\_textQ.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger ParamQ = 0;

if (f10\_textQ.TextLength > 0)

ParamQ = BigInteger.Parse(f10\_textQ.Text);

BigInteger ParamP = 2 \* ParamQ + 1;

f10\_textP.Text = Convert.ToString(ParamP);

f10\_textG.Text = new string(f10\_textG.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger ParamG = 0;

if (f10\_textG.TextLength > 0)

ParamG = BigInteger.Parse(f10\_textG.Text) % ParamP;

f10\_textG.Text = Convert.ToString(ParamG);

if (RabinMiller.IsPrime(ParamQ, 3) && RabinMiller.IsPrime(ParamP, 3) &&

((NumberTheory.BinaryModPow(ParamG, ParamQ, ParamP) != 1)))

{

f10\_textQ.ReadOnly = true;

f10\_textG.ReadOnly = true;

f10\_buttonNextPrimeQ.Enabled = false;

f10\_buttonNextG.Enabled = false;

f10\_chooseHashFun.Enabled = false;

f10\_buttonSetParams.Enabled = false;

f10\_buttonClearParams.Enabled = true;

f10\_button\_GetRandomSecret.Enabled = true;

f10\_button\_GetRandomSessionK.Enabled = true;

f10\_textSessionKey.ReadOnly = false;

f10\_textSecretKey.ReadOnly = false;

f10\_button\_SetSecretKey.Enabled = true;

f10\_button\_ClearSecretKey.Enabled = false;

}

}

private void f10\_button\_SetSecretKey\_Click(object sender, EventArgs e)

{

BigInteger P = BigInteger.Parse(f10\_textP.Text);

BigInteger G = BigInteger.Parse(f10\_textG.Text);

f10\_textSecretKey.Text = new string(f10\_textSecretKey.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger SecretKey = 0;

if (f10\_textSecretKey.TextLength > 0)

SecretKey = BigInteger.Parse(f10\_textSecretKey.Text);

f10\_textSessionKey.Text = new string(f10\_textSessionKey.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger SessionKey = 0;

if (f10\_textSessionKey.TextLength > 0)

SessionKey = BigInteger.Parse(f10\_textSessionKey.Text);

if (1 < SecretKey && SecretKey < P - 1 && 1 < SessionKey && SessionKey < P - 1 &&

NumberTheory.GCD(SessionKey, P - 1, out BigInteger x, out BigInteger y) == 1)

{

f10\_textOpenKey.Text = Convert.ToString(NumberTheory.BinaryModPow(G, SecretKey, P));

f10\_textSecretKey.ReadOnly = true;

f10\_textSessionKey.ReadOnly = true;

f10\_button\_ClearSecretKey.Enabled = true;

f10\_button\_GetRandomSecret.Enabled = false;

f10\_button\_GetRandomSessionK.Enabled = false;

f10\_button\_SetSecretKey.Enabled = false;

f10\_buttonSign.Enabled = true;

}

}

**Результат работы программы:**

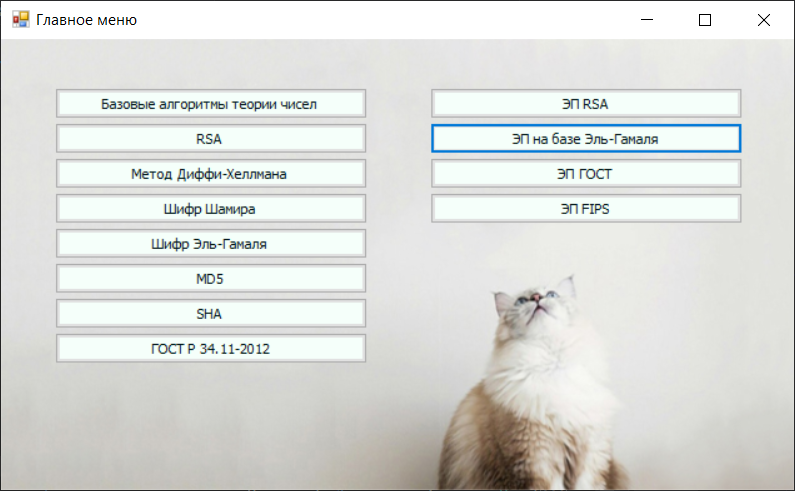


Рисунок 1. Главное меню программы.

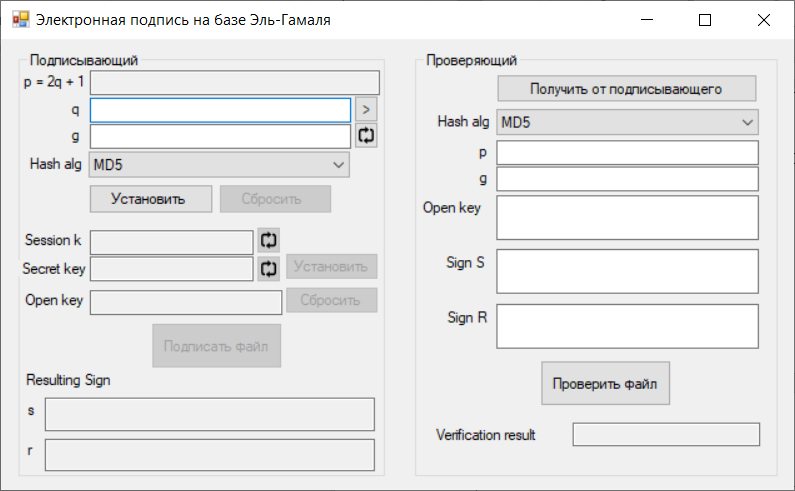


Рисунок 2. Окно электронной подписи на базе шифра Эль-Гамаля.

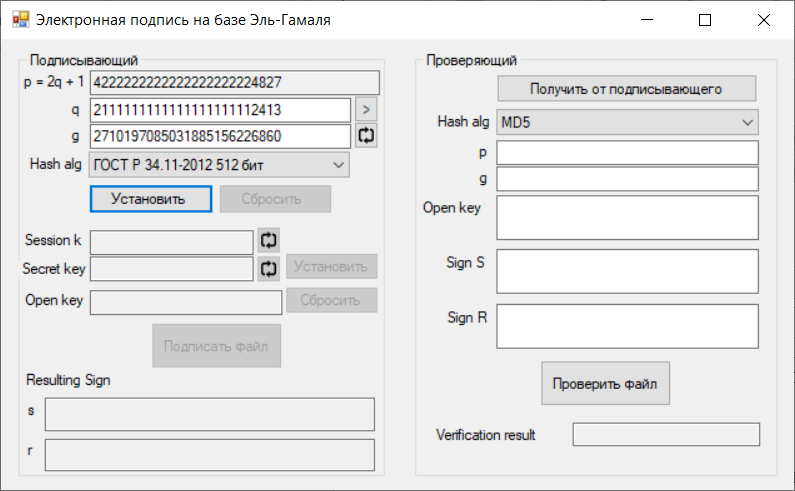


Рисунок 3. Установка параметров системы.

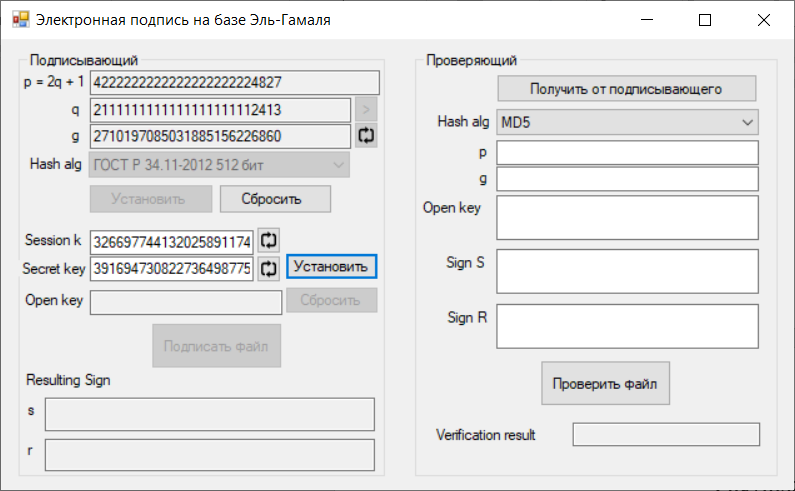


Рисунок 4. Установка ключей.

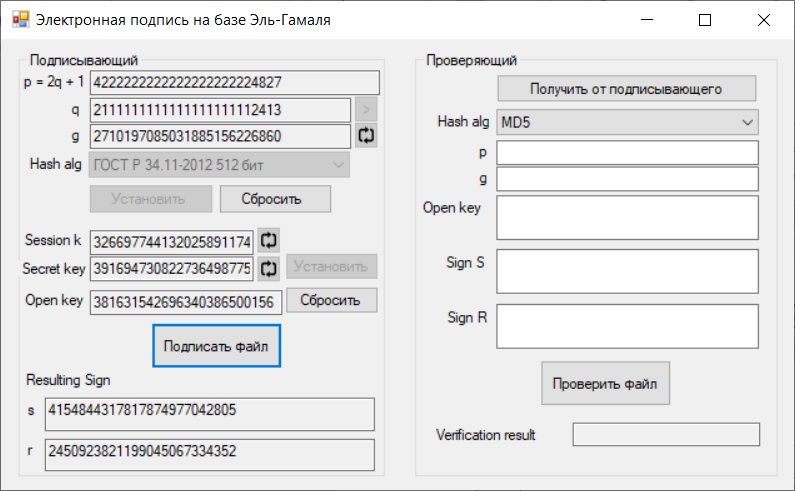


Рисунок 5. Подпись файла с текстом qwerty.

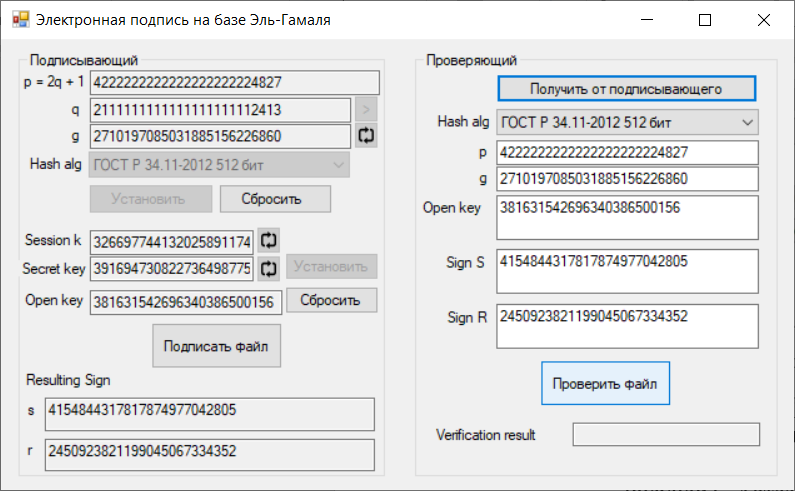


Рисунок 6. Автозаполнение полей для проверки.

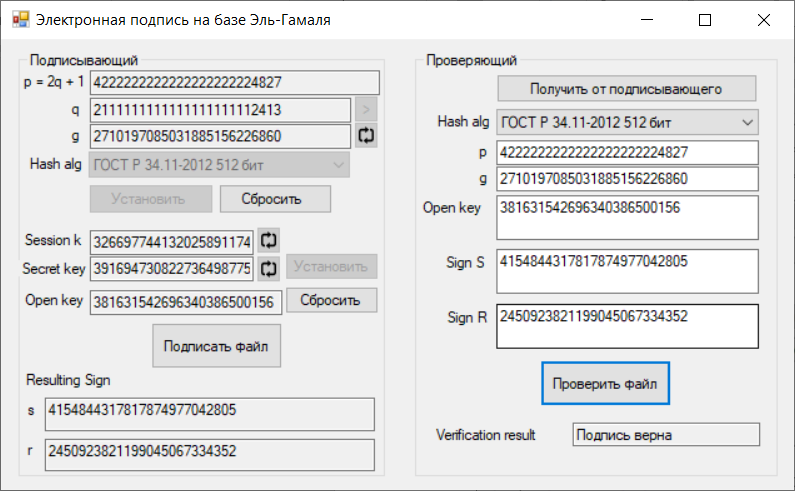


Рисунок 7. Результат проверки файла с текстом qwerty.

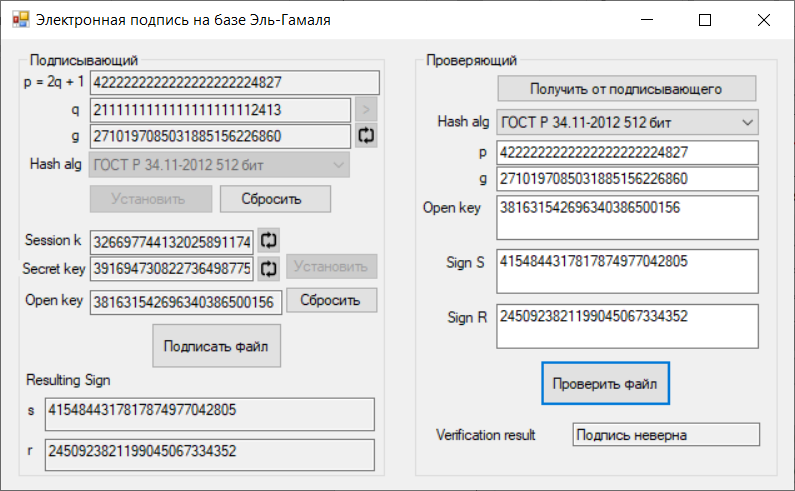


Рисунок 8. Результат проверки файла с другим текстом.

**Вывод:**

В ходе выполнения лабораторной работы я ознакомилась с алгоритмом электронной подписи на базе шифра Эль-Гамаля, реализовала на практике программу подписи и ее проверки. Рассмотренный метод электронной подписи сложнее, чем RSA, а его стойкость базируется на другой, нежели в RSA, односторонней функции.