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 №11**

|  |
| --- |
| Алгоритмы электронной подписи ГОСТ Р 34.10-94 и DSA FIPS-186 |
| *тема* |

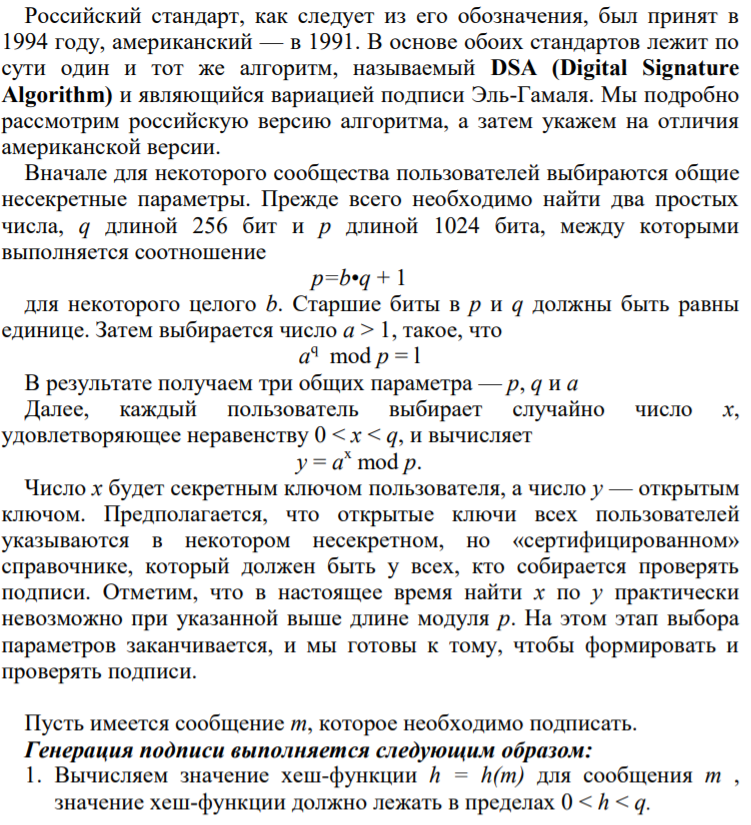
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Руководитель | | |  |  |  | В.И.Вайнштейн |
|  | |  |  | *подпись, дата* |  | *инициалы, фамилия* |
| Студент | КИ15-01 №031508683 | |  |  |  | М.С.Димаксян |
|  | *номер группы, зачетной книжки* | |  | *подпись, дата* |  | *инициалы,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

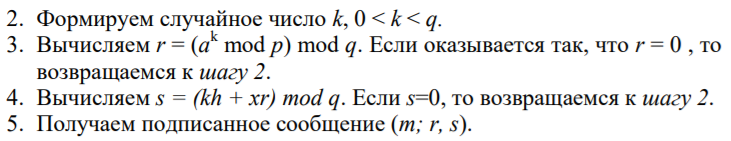
Красноярск 2019

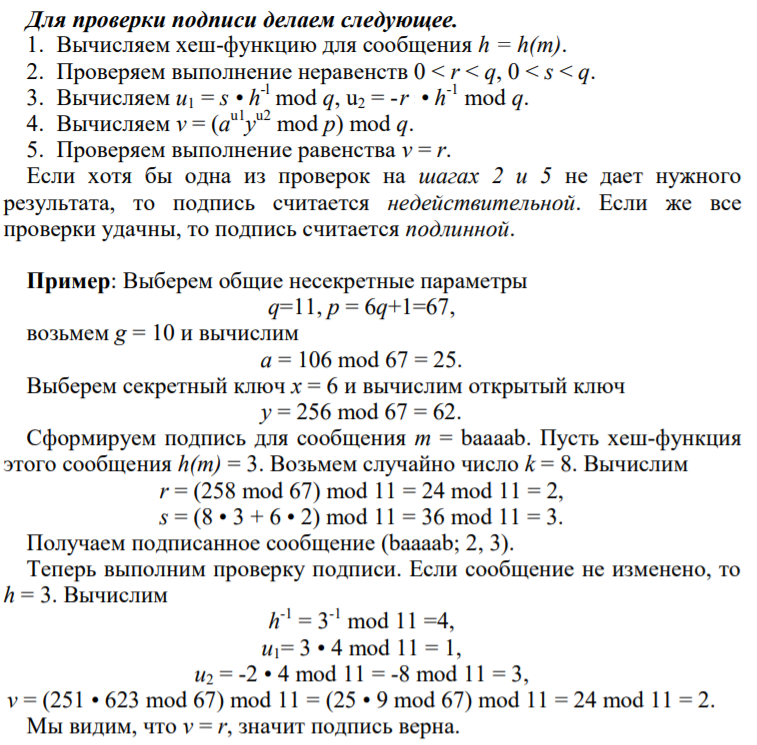
**Задание:**

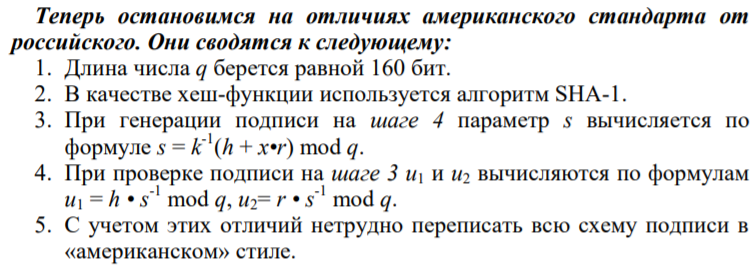
Изучить основные принципы построения и реализации алгоритмов электронной подписи ГОСТ Р 34.10-94 и DSA FIPS-186.

**Теория:**

****

****

****

****

**Исходный код:**

**Функция подписи (ГОСТ):**

private void f11\_buttonSign\_Click(object sender, EventArgs e)

{

BigInteger P = BigInteger.Parse(f11\_textP.Text);

BigInteger Q = BigInteger.Parse(f11\_textQ.Text);

BigInteger A = BigInteger.Parse(f11\_textA.Text);

BigInteger X = BigInteger.Parse(f11\_textSecretKey.Text);

BigInteger K = BigInteger.Parse(f11\_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 (f11\_chooseHashFun.Text == "MD5")

fileHash = HashFunctions.MD5.GetHashMD5(file);

if (f11\_chooseHashFun.Text == "SHA-1")

fileHash = HashFunctions.SHA1.GetHashSHA1(file);

if (f11\_chooseHashFun.Text == "ГОСТ Р 34.11-2012 256 бит")

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

if (f11\_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(A, K, P) % Q;

BigInteger S = (K \* hashDigit + X \* R) % Q;

if (S > 0 && R > 0)

{

f11\_textBoxSign.Text = Convert.ToString(S);

f11\_textBoxSignR.Text = Convert.ToString(R);

//f11\_button\_ClearSecretKey.PerformClick();

//f11\_textSessionKey.Focus();

}

else

{

f11\_textBoxSign.Text = "";

f11\_textBoxSignR.Text = "";

MessageBox.Show("Пожалуйста, выберите другой сессионный ключ", "Ошибка");

f11\_button\_ClearSecretKey.PerformClick();

f11\_textSessionKey.Focus();

}

}

#### Функция проверки подписи (ГОСТ):

private void f11\_buttonCheckSign\_Click(object sender, EventArgs e)

{

f11\_textP\_Recieved.Text = new string(f11\_textP\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

f11\_textQ\_Recieved.Text = new string(f11\_textQ\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

f11\_textA\_Recieved.Text = new string(f11\_textA\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

f11\_textOpenKeyRecieved.Text = new string(f11\_textOpenKeyRecieved.Text.Where(t => char.IsDigit(t)).ToArray());

f11\_textBoxSignR\_Recieved.Text = new string(f11\_textBoxSignR\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

f11\_textBoxSignS\_Recieved.Text = new string(f11\_textBoxSignS\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

if (f11\_textP\_Recieved.TextLength \* f11\_textQ\_Recieved.TextLength \* f11\_textA\_Recieved.TextLength

\* f11\_textOpenKeyRecieved.TextLength \* f11\_textBoxSignR\_Recieved.TextLength \* f11\_textBoxSignS\_Recieved.TextLength != 0)

{

//если хоть одно поле пустое, подпись не проверится

BigInteger P = BigInteger.Parse(f11\_textP\_Recieved.Text);

BigInteger Q = BigInteger.Parse(f11\_textQ\_Recieved.Text);

BigInteger A = BigInteger.Parse(f11\_textA\_Recieved.Text);

BigInteger Y = BigInteger.Parse(f11\_textOpenKeyRecieved.Text);

BigInteger R = BigInteger.Parse(f11\_textBoxSignR\_Recieved.Text);

BigInteger S = BigInteger.Parse(f11\_textBoxSignS\_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 (f11\_chooseHashFun.Text == "MD5")

fileHash = HashFunctions.MD5.GetHashMD5(file);

if (f11\_chooseHashFun.Text == "SHA-1")

fileHash = HashFunctions.SHA1.GetHashSHA1(file);

if (f11\_chooseHashFun.Text == "ГОСТ Р 34.11-2012 256 бит")

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

if (f11\_chooseHashFun.Text == "ГОСТ Р 34.11-2012 512 бит")

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

}

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

BigInteger antiHash = NumberTheory.Foo(hashDigit, Q);

BigInteger u1 = (S \* antiHash) % Q;

BigInteger u2 = (((-R \* antiHash) % Q) + Q) % Q; BigInteger v =

((NumberTheory.BinaryModPow(A, u1, P) \* NumberTheory.BinaryModPow(Y, u2, P)) % P) % Q;

if (0 < S && S < Q && 0 < R && R < Q && v == R)

{

f11\_textVerificationResult.Text = "Подпись верна";

}

else

{

f11\_textVerificationResult.Text = "Подпись неверна";

}

}

#### }

#### Функции установки параметров (ГОСТ):

private void f11\_buttonNextPrimeQ\_Click(object sender, EventArgs e)

{

f11\_textQ.Text = new string(f11\_textQ.Text.Where(t => char.IsDigit(t)).ToArray()); //очищаем Q от ненужных символов

BigInteger ParamQ = 0; //Считываем Q

if (f11\_textQ.TextLength > 0)

ParamQ = BigInteger.Parse(f11\_textQ.Text);

BigInteger minQ = BigInteger.Pow(2, 255);

BigInteger maxQ = BigInteger.Pow(2, 256) - 1;

if (ParamQ < minQ || ParamQ > maxQ)

ParamQ = 0;

if (ParamQ == 0) ParamQ = minQ;

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

while (!RabinMiller.IsPrime(ParamQ, 3))

{

ParamQ += 2;

if (ParamQ > maxQ)

{

ParamQ = minQ;

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

}

}

f11\_textQ.Text = Convert.ToString(ParamQ);

f11\_textP.Text = "";

}

private void f11\_buttonNextPrimeP\_Click(object sender, EventArgs e)

{

f11\_textQ.Text = new string(f11\_textQ.Text.Where(t => char.IsDigit(t)).ToArray());

f11\_textP.Text = new string(f11\_textP.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger ParamQ = 0;

if (f11\_textQ.TextLength > 0)

ParamQ = BigInteger.Parse(f11\_textQ.Text);

BigInteger minQ = BigInteger.Pow(2, 255);

BigInteger maxQ = BigInteger.Pow(2, 256) - 1;

if (ParamQ > minQ && ParamQ < maxQ && RabinMiller.IsPrime(ParamQ, 3))

{

BigInteger ParamP = 0;

if (f11\_textP.TextLength > 0)

ParamP = BigInteger.Parse(f11\_textP.Text);

BigInteger maxP = BigInteger.Pow(2,1024) - 1;

BigInteger minP = BigInteger.Pow(2, 1023);

if (ParamP < minP || ParamP > maxP)

ParamP = 0;

if (ParamP == 0) ParamP = minP;

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

ParamP = ((ParamP / ParamQ) + 1) \* ParamQ + 1;

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

while (!RabinMiller.IsPrime(ParamP, 3))

{

ParamP += 2\*ParamQ;

if (ParamP > maxP)

{

ParamP = minP;

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

ParamP = ((ParamP / ParamQ) + 1) \* ParamQ + 1;

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

}

}

f11\_textQ.Text = Convert.ToString(ParamQ);

f11\_textP.Text = Convert.ToString(ParamP);

}

else

{

f11\_textQ.Text = "";

f11\_textP.Text = "";

}

}

private void f11\_buttonGetRandomA\_Click(object sender, EventArgs e)

{

f11\_textQ.Text = new string(f11\_textQ.Text.Where(t => char.IsDigit(t)).ToArray());

f11\_textP.Text = new string(f11\_textP.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger ParamP = 0;

if (f11\_textP.TextLength > 0)

ParamP = BigInteger.Parse(f11\_textP.Text);

BigInteger ParamQ = 0;

if (f11\_textQ.TextLength > 0)

ParamQ = BigInteger.Parse(f11\_textQ.Text);

if (ParamQ > 0 && ParamP > 0 && ((ParamP - 1) % ParamQ == 0) && RabinMiller.IsPrime(ParamQ, 3) && RabinMiller.IsPrime(ParamP, 3))

{

RandomNumberGenerator rng = RandomNumberGenerator.Create();

BigInteger ParamA = 0;

while (NumberTheory.BinaryModPow(ParamA, ParamQ, ParamP) != 1)

{

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

BigInteger b = (ParamP - 1) / ParamQ;

BigInteger f = NumberTheory.BinaryModPow(d, b, ParamP);

if (f != 1)

{

ParamA = f;

}

}

f11\_textA.Text = Convert.ToString(ParamA);

}

else

{

MessageBox.Show("Сначала подберите допустимые P и Q.");

}

}

private void f11\_buttonSetParams\_Click(object sender, EventArgs e)

{

f11\_textQ.Text = new string(f11\_textQ.Text.Where(t => char.IsDigit(t)).ToArray());

f11\_textP.Text = new string(f11\_textP.Text.Where(t => char.IsDigit(t)).ToArray());

f11\_textA.Text = new string(f11\_textA.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger ParamP = 0;

if (f11\_textP.TextLength > 0)

ParamP = BigInteger.Parse(f11\_textP.Text);

BigInteger ParamQ = 0;

if (f11\_textQ.TextLength > 0)

ParamQ = BigInteger.Parse(f11\_textQ.Text);

BigInteger ParamA = 0;

if (f11\_textA.TextLength > 0)

ParamA = BigInteger.Parse(f11\_textA.Text);

if (ParamP > 0 && ParamQ > 0 && ParamA > 0 && ((ParamP -1) % ParamQ == 0) && NumberTheory.BinaryModPow(ParamA, ParamQ, ParamP) == 1

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

{

f11\_textP.ReadOnly = true;

f11\_textQ.ReadOnly = true;

f11\_textA.ReadOnly = true;

f11\_buttonNextPrimeQ.Enabled = false;

f11\_buttonNextPrimeP.Enabled = false;

f11\_buttonGetRandomA.Enabled = false;

f11\_chooseHashFun.Enabled = false;

f11\_buttonSetParams.Enabled = false;

f11\_buttonClearParams.Enabled = true;

f11\_textSessionKey.ReadOnly = false;

f11\_textSecretKey.ReadOnly = false;

f11\_button\_GetRandomSessionK.Enabled = true;

f11\_button\_GetRandomSecret.Enabled = true;

f11\_button\_SetSecretKey.Enabled = true;

f11\_button\_ClearSecretKey.Enabled = false;//в случае сброса тоже

f11\_buttonSign.Enabled = false;//в случае сброса тоже

}

}

private void f11\_button\_GetRandomSecret\_Click(object sender, EventArgs e)

{

RandomNumberGenerator rng = RandomNumberGenerator.Create();

BigInteger Q = BigInteger.Parse(f11\_textQ.Text);

BigInteger A = BigInteger.Parse(f11\_textA.Text);

BigInteger P = BigInteger.Parse(f11\_textP.Text);

BigInteger x = BigIntegerRandomUtils.RandomInRange(rng, 2, Q - 1);

BigInteger y = NumberTheory.BinaryModPow(A, x, P);

f11\_textSecretKey.Text = Convert.ToString(x);

f11\_textOpenKey.Text = Convert.ToString(y);

}

private void f11\_button\_GetRandomSessionK\_Click(object sender, EventArgs e)

{

RandomNumberGenerator rng = RandomNumberGenerator.Create();

BigInteger Q = BigInteger.Parse(f11\_textQ.Text);

BigInteger A = BigInteger.Parse(f11\_textA.Text);

BigInteger P = BigInteger.Parse(f11\_textP.Text);

BigInteger k = BigIntegerRandomUtils.RandomInRange(rng, 2, Q - 1);

f11\_textSessionKey.Text = Convert.ToString(k);

}

private void f11\_button\_SetSecretKey\_Click(object sender, EventArgs e)

{

BigInteger Q = BigInteger.Parse(f11\_textQ.Text);

BigInteger A = BigInteger.Parse(f11\_textA.Text);

BigInteger P = BigInteger.Parse(f11\_textP.Text);

f11\_textSecretKey.Text = new string(f11\_textSecretKey.Text.Where(t => char.IsDigit(t)).ToArray());

f11\_textSessionKey.Text = new string(f11\_textSessionKey.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger secret = 0;

if (f11\_textSecretKey.TextLength > 0)

{

secret = BigInteger.Parse(f11\_textSecretKey.Text) % P;

f11\_textSecretKey.Text = Convert.ToString(secret);

}

BigInteger session = 0;

if (f11\_textSessionKey.TextLength > 0)

{

session = BigInteger.Parse(f11\_textSessionKey.Text) % P;

f11\_textSessionKey.Text = Convert.ToString(session);

}

BigInteger y = NumberTheory.BinaryModPow(A, secret, P);

f11\_textOpenKey.Text = Convert.ToString(y);

if (1 < session && session < Q && 1 < secret && secret < Q)

{

f11\_textSessionKey.ReadOnly = true;

f11\_textSecretKey.ReadOnly = true;

f11\_button\_GetRandomSessionK.Enabled = false;

f11\_button\_GetRandomSecret.Enabled = false;

f11\_button\_SetSecretKey.Enabled = false;

f11\_button\_ClearSecretKey.Enabled = true;

f11\_buttonSign.Enabled = true;

}

}

**Функция подписи (FIPS):**

private void f12\_buttonSign\_Click(object sender, EventArgs e)

{

BigInteger P = BigInteger.Parse(f12\_textP.Text);

BigInteger Q = BigInteger.Parse(f12\_textQ.Text);

BigInteger A = BigInteger.Parse(f12\_textA.Text);

BigInteger X = BigInteger.Parse(f12\_textSecretKey.Text);

BigInteger K = BigInteger.Parse(f12\_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 (f12\_chooseHashFun.Text == "MD5")

fileHash = HashFunctions.MD5.GetHashMD5(file);

if (f12\_chooseHashFun.Text == "SHA-1")

fileHash = HashFunctions.SHA1.GetHashSHA1(file);

if (f12\_chooseHashFun.Text == "ГОСТ Р 34.11-2012 256 бит")

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

if (f12\_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(A, K, P) % Q;

BigInteger antiK = NumberTheory.Foo(K, Q);

BigInteger S = (antiK \*(hashDigit + X\*R)) % Q;

if (S > 0 && R > 0)

{

f12\_textBoxSign.Text = Convert.ToString(S);

f12\_textBoxSignR.Text = Convert.ToString(R);

//f12\_button\_ClearSecretKey.PerformClick();

//f12\_textSessionKey.Focus();

}

else

{

f12\_textBoxSign.Text = "";

f12\_textBoxSignR.Text = "";

MessageBox.Show("Пожалуйста, выберите другой сессионный ключ", "Ошибка");

f12\_button\_ClearSecretKey.PerformClick();

f12\_textSessionKey.Focus();

}

}

#### Функция проверки подписи (FIPS):

private void f12\_buttonCheckSign\_Click(object sender, EventArgs e)

{

f12\_textP\_Recieved.Text = new string(f12\_textP\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

f12\_textQ\_Recieved.Text = new string(f12\_textQ\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

f12\_textA\_Recieved.Text = new string(f12\_textA\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

f12\_textOpenKeyRecieved.Text = new string(f12\_textOpenKeyRecieved.Text.Where(t => char.IsDigit(t)).ToArray());

f12\_textBoxSignR\_Recieved.Text = new string(f12\_textBoxSignR\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

f12\_textBoxSignS\_Recieved.Text = new string(f12\_textBoxSignS\_Recieved.Text.Where(t => char.IsDigit(t)).ToArray());

if (f12\_textP\_Recieved.TextLength \* f12\_textQ\_Recieved.TextLength \* f12\_textA\_Recieved.TextLength

\* f12\_textOpenKeyRecieved.TextLength \* f12\_textBoxSignR\_Recieved.TextLength \* f12\_textBoxSignS\_Recieved.TextLength != 0)

{

//если хоть одно поле пустое, подпись не проверится

BigInteger P = BigInteger.Parse(f12\_textP\_Recieved.Text);

BigInteger Q = BigInteger.Parse(f12\_textQ\_Recieved.Text);

BigInteger A = BigInteger.Parse(f12\_textA\_Recieved.Text);

BigInteger Y = BigInteger.Parse(f12\_textOpenKeyRecieved.Text);

BigInteger R = BigInteger.Parse(f12\_textBoxSignR\_Recieved.Text);

BigInteger S = BigInteger.Parse(f12\_textBoxSignS\_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 (f12\_chooseHashFun.Text == "MD5")

fileHash = HashFunctions.MD5.GetHashMD5(file);

if (f12\_chooseHashFun.Text == "SHA-1")

fileHash = HashFunctions.SHA1.GetHashSHA1(file);

if (f12\_chooseHashFun.Text == "ГОСТ Р 34.11-2012 256 бит")

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

if (f12\_chooseHashFun.Text == "ГОСТ Р 34.11-2012 512 бит")

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

}

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

BigInteger antiS = NumberTheory.Foo(S, Q);

BigInteger u1 = (hashDigit \* antiS) % Q;

BigInteger u2 = (R \* antiS) % Q;

BigInteger v =

((NumberTheory.BinaryModPow(A, u1, P) \* NumberTheory.BinaryModPow(Y, u2, P)) % P) % Q;

if (0 < S && S < Q && 0 < R && R < Q && v == R)

{

f12\_textVerificationResult.Text = "Подпись верна";

}

else

{

f12\_textVerificationResult.Text = "Подпись неверна";

}

}

#### }

#### Функции установки параметров (FIPS):

private void f12\_buttonNextPrimeQ\_Click(object sender, EventArgs e)

{

f12\_textQ.Text = new string(f12\_textQ.Text.Where(t => char.IsDigit(t)).ToArray()); //очищаем Q от ненужных символов

BigInteger ParamQ = 0; //Считываем Q

if (f12\_textQ.TextLength > 0)

ParamQ = BigInteger.Parse(f12\_textQ.Text);

BigInteger minQ = BigInteger.Pow(2, 159);

BigInteger maxQ = BigInteger.Pow(2, 160) - 1;

if (ParamQ < minQ || ParamQ > maxQ)

ParamQ = 0;

if (ParamQ == 0) ParamQ = minQ;

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

while (!RabinMiller.IsPrime(ParamQ, 3))

{

ParamQ += 2;

if (ParamQ > maxQ)

{

ParamQ = minQ;

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

}

}

f12\_textQ.Text = Convert.ToString(ParamQ);

f12\_textP.Text = "";

}

private void f12\_buttonNextPrimeP\_Click(object sender, EventArgs e)

{

f12\_textQ.Text = new string(f12\_textQ.Text.Where(t => char.IsDigit(t)).ToArray());

f12\_textP.Text = new string(f12\_textP.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger ParamQ = 0;

if (f12\_textQ.TextLength > 0)

ParamQ = BigInteger.Parse(f12\_textQ.Text);

BigInteger minQ = BigInteger.Pow(2, 159);

BigInteger maxQ = BigInteger.Pow(2, 160) - 1;

if (ParamQ > minQ && ParamQ < maxQ && RabinMiller.IsPrime(ParamQ, 3))

{

BigInteger ParamP = 0;

if (f12\_textP.TextLength > 0)

ParamP = BigInteger.Parse(f12\_textP.Text);

BigInteger maxP = BigInteger.Pow(2, 1024) - 1;

BigInteger minP = BigInteger.Pow(2, 1023);

if (ParamP < minP || ParamP > maxP)

ParamP = 0;

if (ParamP == 0) ParamP = minP;

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

ParamP = ((ParamP / ParamQ) + 1) \* ParamQ + 1;

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

while (!RabinMiller.IsPrime(ParamP, 3))

{

ParamP += 2 \* ParamQ;

if (ParamP > maxP)

{

ParamP = minP;

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

ParamP = ((ParamP / ParamQ) + 1) \* ParamQ + 1;

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

}

}

f12\_textQ.Text = Convert.ToString(ParamQ);

f12\_textP.Text = Convert.ToString(ParamP);

}

else

{

f12\_textQ.Text = "";

f12\_textP.Text = "";

}

}

private void f12\_buttonGetRandomA\_Click(object sender, EventArgs e)

{

f12\_textQ.Text = new string(f12\_textQ.Text.Where(t => char.IsDigit(t)).ToArray());

f12\_textP.Text = new string(f12\_textP.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger ParamP = 0;

if (f12\_textP.TextLength > 0)

ParamP = BigInteger.Parse(f12\_textP.Text);

BigInteger ParamQ = 0;

if (f12\_textQ.TextLength > 0)

ParamQ = BigInteger.Parse(f12\_textQ.Text);

if (ParamQ > 0 && ParamP > 0 && ((ParamP - 1) % ParamQ == 0) && RabinMiller.IsPrime(ParamQ, 3) && RabinMiller.IsPrime(ParamP, 3))

{

RandomNumberGenerator rng = RandomNumberGenerator.Create();

BigInteger ParamA = 0;

while (NumberTheory.BinaryModPow(ParamA, ParamQ, ParamP) != 1)

{

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

BigInteger b = (ParamP - 1) / ParamQ;

BigInteger f = NumberTheory.BinaryModPow(d, b, ParamP);

if (f != 1)

{

ParamA = f;

}

}

f12\_textA.Text = Convert.ToString(ParamA);

}

else

{

MessageBox.Show("Сначала подберите допустимые P и Q.");

}

}

private void f12\_buttonSetParams\_Click(object sender, EventArgs e)

{

f12\_textQ.Text = new string(f12\_textQ.Text.Where(t => char.IsDigit(t)).ToArray());

f12\_textP.Text = new string(f12\_textP.Text.Where(t => char.IsDigit(t)).ToArray());

f12\_textA.Text = new string(f12\_textA.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger ParamP = 0;

if (f12\_textP.TextLength > 0)

ParamP = BigInteger.Parse(f12\_textP.Text);

BigInteger ParamQ = 0;

if (f12\_textQ.TextLength > 0)

ParamQ = BigInteger.Parse(f12\_textQ.Text);

BigInteger ParamA = 0;

if (f12\_textA.TextLength > 0)

ParamA = BigInteger.Parse(f12\_textA.Text);

if (ParamP > 0 && ParamQ > 0 && ParamA > 0 && ((ParamP - 1) % ParamQ == 0) && NumberTheory.BinaryModPow(ParamA, ParamQ, ParamP) == 1

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

{

f12\_textP.ReadOnly = true;

f12\_textQ.ReadOnly = true;

f12\_textA.ReadOnly = true;

f12\_buttonNextPrimeQ.Enabled = false;

f12\_buttonNextPrimeP.Enabled = false;

f12\_buttonGetRandomA.Enabled = false;

f12\_chooseHashFun.Enabled = false;

f12\_buttonSetParams.Enabled = false;

f12\_buttonClearParams.Enabled = true;

f12\_textSessionKey.ReadOnly = false;

f12\_textSecretKey.ReadOnly = false;

f12\_button\_GetRandomSessionK.Enabled = true;

f12\_button\_GetRandomSecret.Enabled = true;

f12\_button\_SetSecretKey.Enabled = true;

f12\_button\_ClearSecretKey.Enabled = false;//в случае сброса тоже

f12\_buttonSign.Enabled = false;//в случае сброса тоже

}

}

private void f12\_button\_GetRandomSecret\_Click(object sender, EventArgs e)

{

RandomNumberGenerator rng = RandomNumberGenerator.Create();

BigInteger Q = BigInteger.Parse(f12\_textQ.Text);

BigInteger A = BigInteger.Parse(f12\_textA.Text);

BigInteger P = BigInteger.Parse(f12\_textP.Text);

BigInteger x = BigIntegerRandomUtils.RandomInRange(rng, 2, Q - 1);

BigInteger y = NumberTheory.BinaryModPow(A, x, P);

f12\_textSecretKey.Text = Convert.ToString(x);

f12\_textOpenKey.Text = Convert.ToString(y);

}

private void f12\_button\_GetRandomSessionK\_Click(object sender, EventArgs e)

{

RandomNumberGenerator rng = RandomNumberGenerator.Create();

BigInteger Q = BigInteger.Parse(f12\_textQ.Text);

BigInteger A = BigInteger.Parse(f12\_textA.Text);

BigInteger P = BigInteger.Parse(f12\_textP.Text);

BigInteger k = BigIntegerRandomUtils.RandomInRange(rng, 2, Q - 1);

f12\_textSessionKey.Text = Convert.ToString(k);

}

private void f12\_button\_SetSecretKey\_Click(object sender, EventArgs e)

{

BigInteger Q = BigInteger.Parse(f12\_textQ.Text);

BigInteger A = BigInteger.Parse(f12\_textA.Text);

BigInteger P = BigInteger.Parse(f12\_textP.Text);

f12\_textSecretKey.Text = new string(f12\_textSecretKey.Text.Where(t => char.IsDigit(t)).ToArray());

f12\_textSessionKey.Text = new string(f12\_textSessionKey.Text.Where(t => char.IsDigit(t)).ToArray());

BigInteger secret = 0;

if (f12\_textSecretKey.TextLength > 0)

{

secret = BigInteger.Parse(f12\_textSecretKey.Text) % P;

f12\_textSecretKey.Text = Convert.ToString(secret);

}

BigInteger session = 0;

if (f12\_textSessionKey.TextLength > 0)

{

session = BigInteger.Parse(f12\_textSessionKey.Text) % P;

f12\_textSessionKey.Text = Convert.ToString(session);

}

BigInteger y = NumberTheory.BinaryModPow(A, secret, P);

f12\_textOpenKey.Text = Convert.ToString(y);

if (1 < session && session < Q && 1 < secret && secret < Q)

{

f12\_textSessionKey.ReadOnly = true;

f12\_textSecretKey.ReadOnly = true;

f12\_button\_GetRandomSessionK.Enabled = false;

f12\_button\_GetRandomSecret.Enabled = false;

f12\_button\_SetSecretKey.Enabled = false;

f12\_button\_ClearSecretKey.Enabled = true;

f12\_buttonSign.Enabled = true;

}

}

**Результат работы программы:**

#### ГОСТ 34.10-94

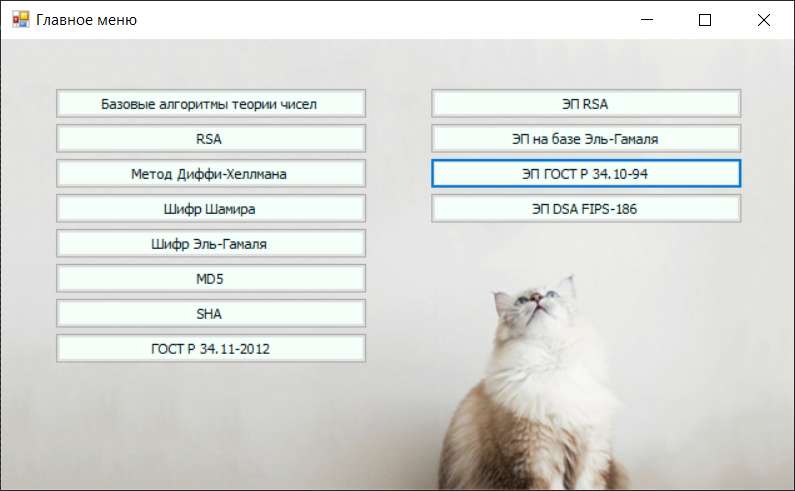


Рисунок 1.1. Главное меню программы.

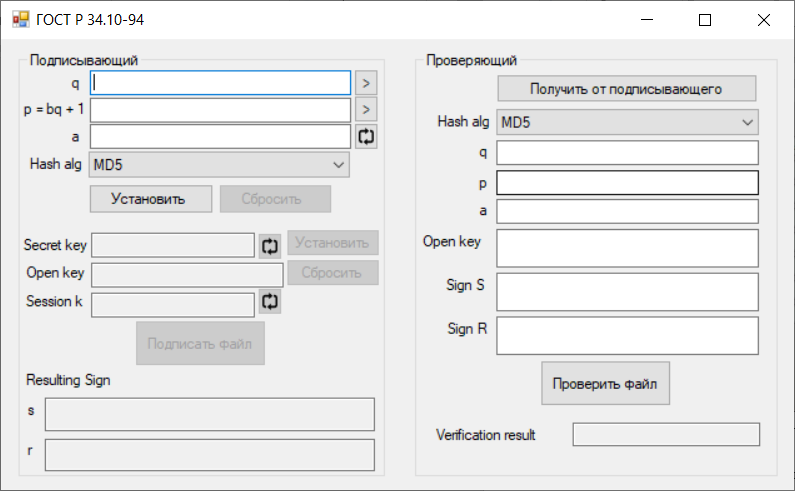


Рисунок 1.2. Окно электронной подписи ГОСТ 34.10-94.

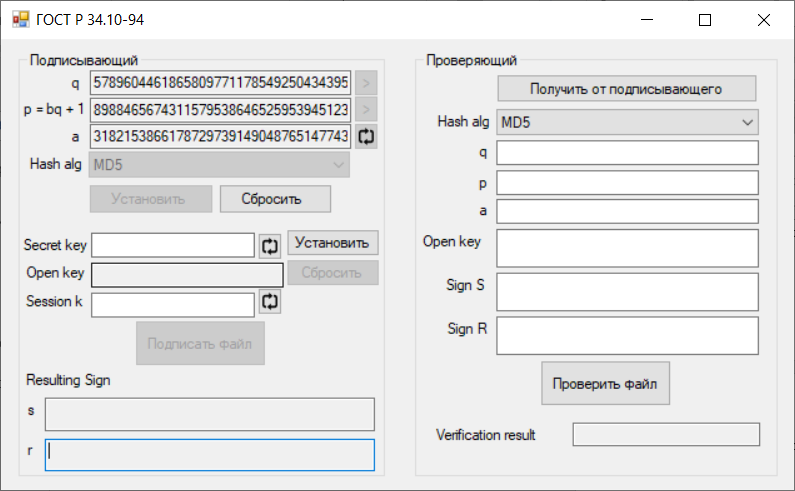


Рисунок 1.3. Установка параметров системы.

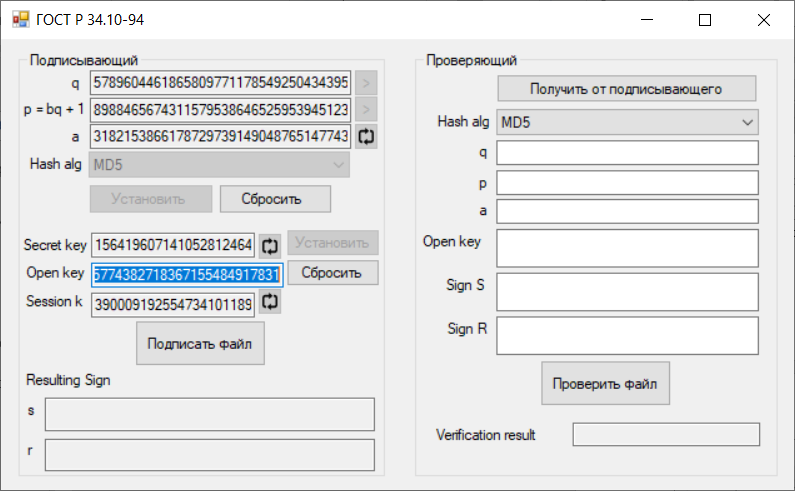


Рисунок 1.4. Установка ключей.

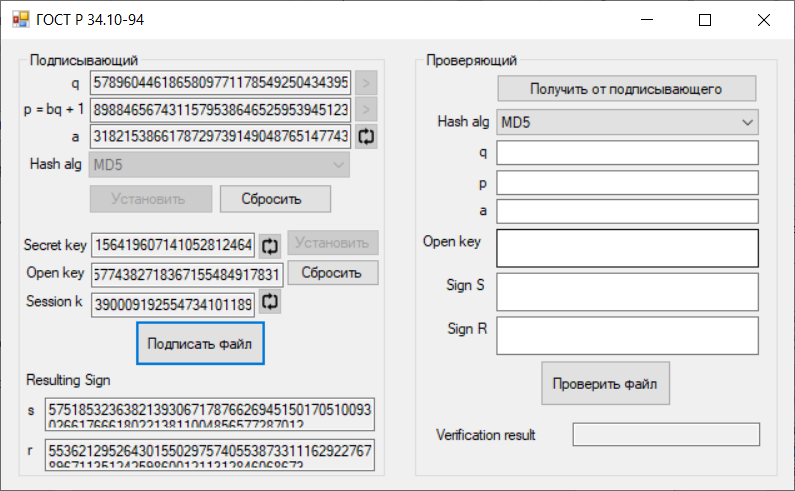


Рисунок 1.5. Подпись файла с текстом qwerty.

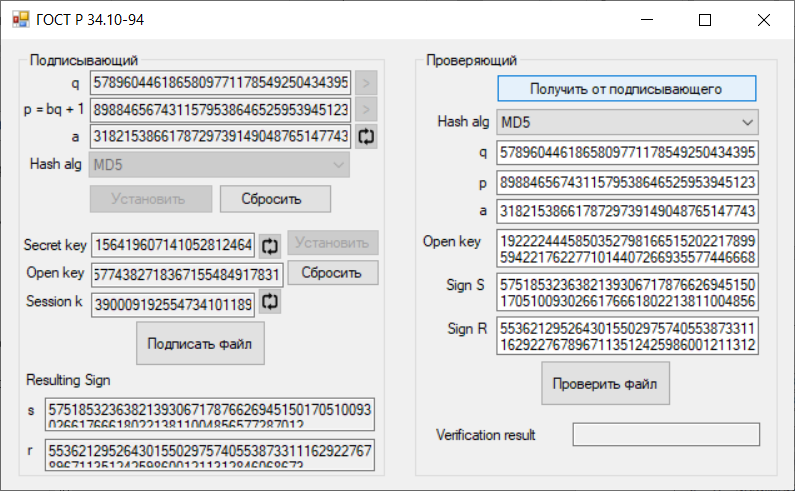


Рисунок 1.6. Автозаполнение полей для проверки.

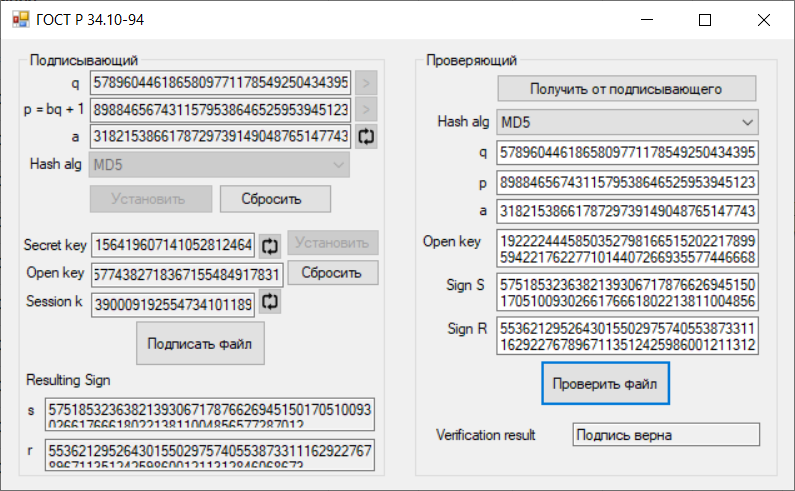


Рисунок 1.7. Результат проверки файла с текстом qwerty.

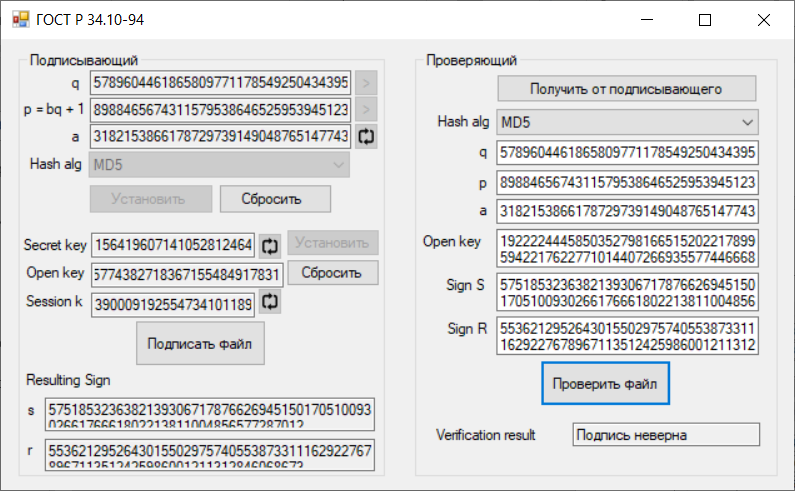


Рисунок 1.8. Результат проверки файла с другим текстом.

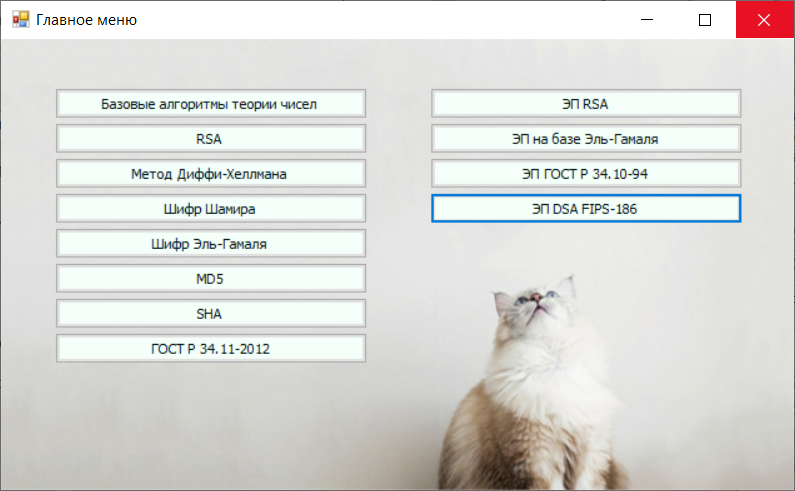


Рисунок 2.1. Главное меню программы.

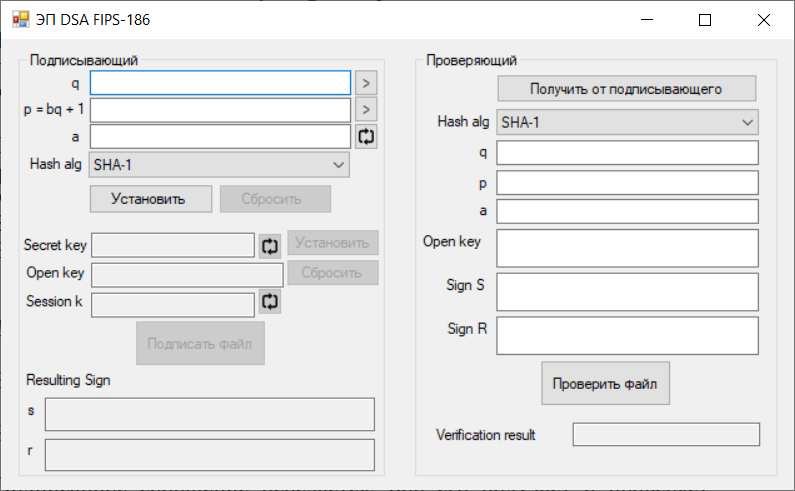


Рисунок 2.2. Окно электронной подписи ГОСТ 34.10-94.

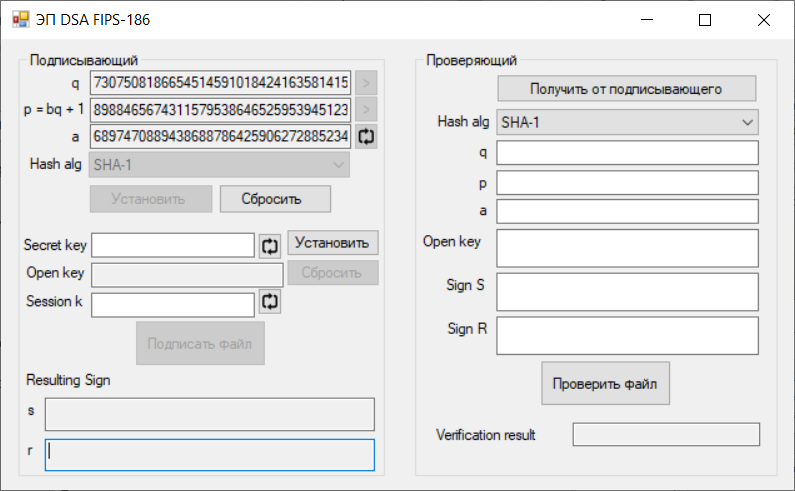


Рисунок 2.3. Установка параметров системы.

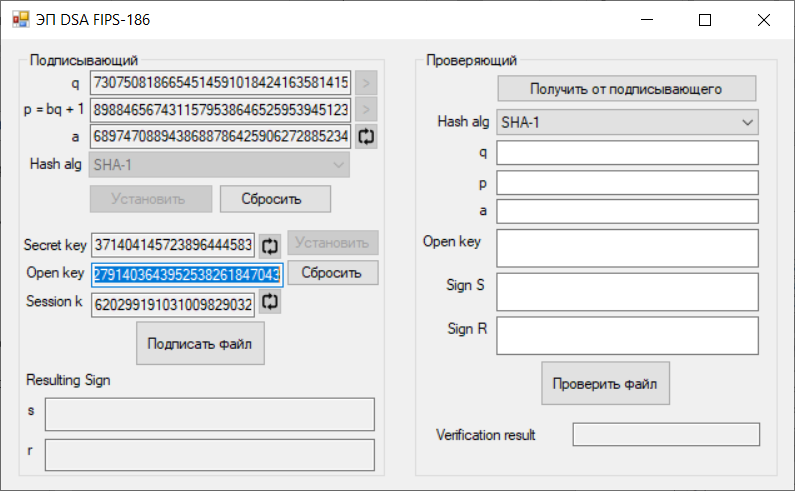


Рисунок 2.4. Установка ключей.

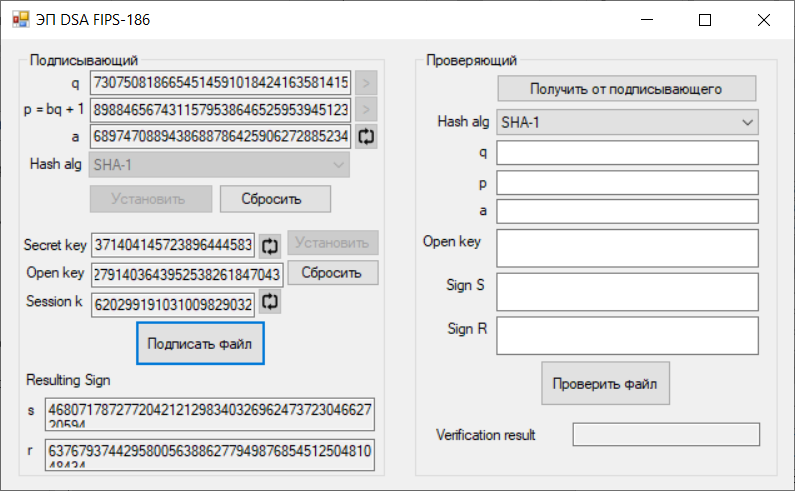


Рисунок 2.5. Подпись файла с текстом qwerty.

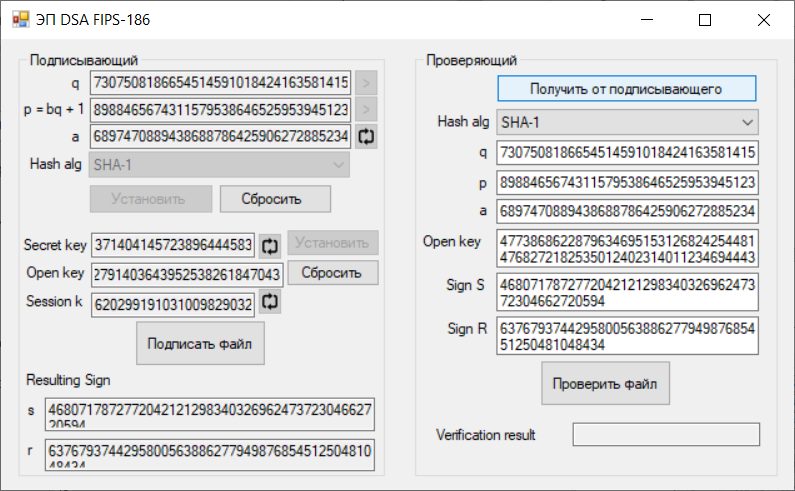


Рисунок 2.6. Автозаполнение полей для проверки.

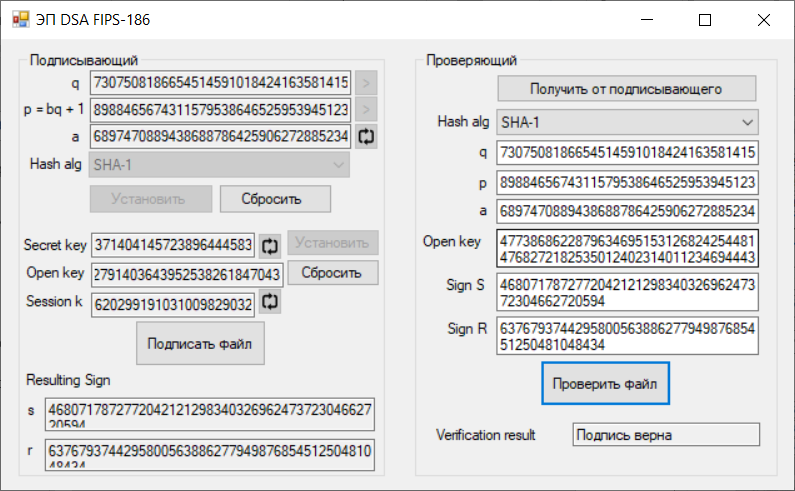


Рисунок 2.7. Результат проверки файла с текстом qwerty.

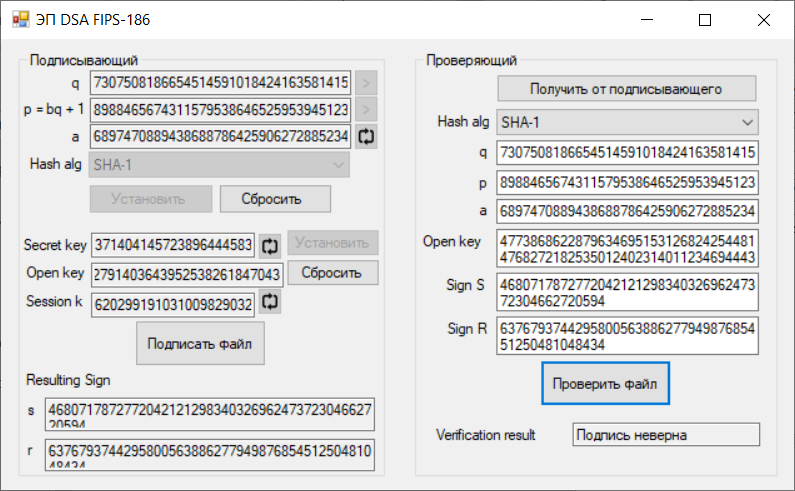


Рисунок 2.8. Результат проверки файла с другим текстом.

**Вывод:**

В ходе выполнения лабораторной работы я ознакомилась с алгоритмами электронной подписи ГОСТ Р 34.10-94 и DSA FIPS-186, реализовала на практике программы подписи и ее проверки. Рассмотренные методы электронной подписи имеют незначительные отличия в работе.