НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«МЭИ»
Институт Автоматики и Вычислительной Техники
Кафедра Прикладной Математики и искусственного интеллекта

Тема: «Программная реализация протокола электронной подписи Шнорра.»

Выполнила: Ростовых А.Д.

Москва, 2022

Оглавление

Введение

В настоящее время основу обеспечения безопасности электронного документооборота составляют системы электронной цифровой подписи (ЭЦП).

ЭЦП – это информация в электронной форме, которая присоединена к другой информации в электронной форме (подписываемой информации) или иным образом связана с такой информацией и которая используется для определения лица, подписывающего информацию. Подпись связана как с автором, так и с самим документом с помощью криптографических методов, и не может быть подделана с помощью обычного копирования. По своему существу электронная подпись представляет собой реквизит электронного документа, позволяющий установить отсутствие искажения информации в электронном документе с момента формирования ЭЦП и проверить принадлежность подписи владельцу сертификата ключа ЭЦП. Значение реквизита получается в результате криптографического преобразования информации с использованием закрытого ключа ЭЦП.

Системы ЭЦП основаны на криптографических алгоритмах. Одним из таких алгоритмов является алгоритм К. Шнорра, рассматриваемый в данной работе. Его надежность основана на практической сложности задачи дискретного логарифмирования. Большая часть вычислений, нужных для генерации подписи и независящих от подписываемого сообщения, может быть выполнена на стадии предварительных вычислений. Эти вычисления могут быть выполнены во время простоя процессора и позволяют увеличить скорость подписания.

В криптографии подпись Шнорра – это цифровая подпись, созданная алгоритмом подписи Шнорра, описанным Клаусом Шнорром. Эта схема цифровой подписи известна своей простотой, эффективностью и генерирует короткие подписи. Это один из протоколов, используемых для реализации доказательства нулевого знания. В криптографии доказательство знания – это интерактивное доказательство, в котором доказывающему удается «убедить» проверяющего в том, что он знает определенную информацию. Для машины знание определяется в терминах вычислений. Машина знает «X», если этот «X» можно вычислить. Проверяющий принимает или отклоняет доказательство. Доказательство подписи должно убедить проверяющего, что они общаются с пользователем, которому известен закрытый ключ, соответствующий открытому ключу.

Целью работы является разработка приложения для проверки и создания электронной цифровой подписи на основе алгоритма Шнорра.

1. Описание протокола

Общепризнанная схема цифровой подписи охватывает три процесса:

1. **Генерация ключевой пары**. При помощи алгоритма генерации ключа равновероятным образом из набора возможных закрытых ключей выбирается закрытый ключ, вычисляется соответствующий ему открытый ключ.
2. **Формирование подписи**. Для заданного электронного документа с помощью закрытого ключа вычисляется подпись.
3. **Проверка (верификация) подписи**. Для данных документа и подписи с помощью открытого ключа определяется действительность подписи. Для того чтобы использование цифровой подписи имело смысл, необходимо выполнение двух условий:
    - Верификация подписи должна производиться открытым ключом, соответствующим именно тому закрытому ключу, который использовался при подписании.
    - Без обладания закрытым ключом должно быть вычислительно сложно создать легитимную цифровую подпись.

В протоколе имеются два участника – Алиса, которая хочет подтвердить свой идентификатор, и Боб, который должен проверить это подтверждение. У Алисы имеется два ключа – V, открытый (общедоступный), и S – закрытый (приватный) ключ, известный только Алисе. Фактически, Боб должен проверить, что Алиса знает свой закрытый ключ S, используя только V.

1.1. Генерация пары ключей

1. Выбирается простое число p (обычно равняется 512 или 1024 битам)
2. Выбирается другое простое число q (p-1 должно делиться на q, другими словами должно выполняться p-1= 0 {mod q}, q обычно выбирают равным 160 битам).
3. Выбирается число $a \neq 1$ (такое, что $a^q=1$ {mod p}).
   a, p и q – открытые параметры.
4. Выбирается закрытый ключ s, такой, что $s<q$.
5. Вычисляется открытый ключ $v=a^{q-s}$ {mod p}.

Открытый ключ “v” будет глобальным и общедоступным наряду с p, q и a. Однако только Алиса будет знать закрытый ключ "s".

1.2. Формирование подписи

Теперь Алиса создает подпись и хочет отправить зашифрованное сообщение "M" Бобу. Требуется выполнить следующие шаги, чтобы использовать подпись Шнорра:

1. Сначала Алиса должна выбрать случайное число “r”, такое, что $0<r<q$.
2. Вычислить значение X такое, что: $X= a^r \{mod\ p\}$
   Это стадия предварительных вычислений. Стоит отметить, что для подписи разных сообщений могут использоваться одинаковые открытый и закрытый ключи, в то время как число r выбирается заново для каждого сообщения.
3. Теперь, когда Алиса вычислила значение X, требуется объединить это с исходным сообщением (так же, как конкатенация строк). Итак, она собирается объединить M и X, чтобы получить M||X. и она собирается сохранить хэш этого значения в e:
   e = H(M||X), где H() - хэш-функция
4. Затем требуется вычислить вторую подпись – значение “y” такое, что:
   y = (r + s*e) mod q
5. Теперь, когда все вычисления закончены, Алиса собирается отправить следующее Бобу.
   - Сообщение M.
   - Подписи e и y.

   Наряду с этим, Боб имеет следующую общедоступную информацию:-
   - Открытый ключ Алисы – v.
   - Простое число, которое Алиса выбрала – p.
   - q, который является фактором p-1, который выбрала Алиса.
   - a такое, что $a^q = 1 \{mod\ p\}$, выбранное Алисой.

1.3. Проверка подписи

Теперь Боб должен вычислить и проверить X’таким образом, что:

$X' = a^y * v^e \{mod\ p\}$

Если s – правильный закрытый ключ, то:

Известно, что $v = a^{-s} \{mod\ p\}$, при подстановке этого в уравнение выше получается:

$X' = a^y * a^{-se} = a^{(y-s*e)} \{mod\ p\}$

Так же известно, что,

$y = (r + s*e) \bmod q$

Что означает:

$r = (y - s*e) \bmod q$

При подстановке этого значения в уравнение выше:

$X' = a^r \{\bmod p\}$

Как уже было сказано выше: $X = a^r \{\bmod p\}$ и, следовательно $X' = X$

Но Боб не знает значения X, потому что он никогда не получал это значение. Все, что он получил, это следующее: сообщение M, подписи (e и y) и множество открытых переменных (открытый ключ v, p, q и a).

Таким образом, он собирается выполнить проверку для e, выполнив следующее:

$e = H (M||X')$

При этом ранее e было вычислено как:

$e = H(M||X)$

Итак, по этой логике, если два значения e совпадают, это означает, что

$X = X'$

Это следует всем трем свойствам доказательства нулевого знания :

- Полнота – Боб был убежден в честности Алисы, потому что в конце $X = X'$.
- Надежность – план был разумным, потому что у Алисы был только один способ доказать свою честность, и это можно было сделать через ее закрытый ключ.
- Нулевые знания – Боб так и не узнал о закрытом ключе Алисы.

2. Описание программы

2.1. Описание программной реализации

Приложение реализовано на языке C# с использованием технологии Windows Forms .NET Framework.

Логика генерации ключевой пары и формирования подписи заключена в функции *private void signbutton_Click(object sender, EventArgs e).*Здесь происходит проверка правильности введенных чисел с помощью функций:

1. *private bool IsTheNumberSimple(long n)* – проверка числа на простоту.
2. *private bool IsTheDivider(long p, long q)* – проверка, что p-1 делится на q.
3. *private bool IsOKa(long a, long q,long p)* – проверка, что $a^q=1$ {mod p}
4. *private bool IsOKw(long w,long q)* – проверка, что закрытый ключ w<q.

Здесь же происходит вычисление открытого ключа v, а так же ввод числа r и вычисление x по предоставленному в предыдущем пункте алгоритму.

Ввод параметров может осуществлять как вручную, так и с помощью специальных элементов управления (см. п.2.2. «Описание интерфейса»), для чего предназначены следующие функции:

1. *private void Rand_Click(object sender, EventArgs e)* – подбор числа q.
2. *private void RandAV_Click(object sender, EventArgs e)* – подбор числа a.
3. *private void CalcV_Click(object sender, EventArgs e)* – вычисление открытого ключа v.

Для объединения сообщения и числа x используется метод Concat(string, string) библиотеки string языка C#. Хэш вычисляется с помощью метода Object.GetHashCode() , возвращающего хэш-код для заданной строки.

Здесь же происходит вычисление обеих подписей e и y, а так же сохранение шагов протокола в выбранный пользователем файл.

Проверка подписи реализована в функции *private void checkbutton_Click(object sender, EventArgs e)* и осуществляется по описанным шагам протокола. Так как используются большие числа и объемные вычисления, работа происходит с типом BigInteger, представляющим произвольно большое целое число.

2.2. Описание интерфейса

При запуске программы перед пользователем открывается следующее окно:

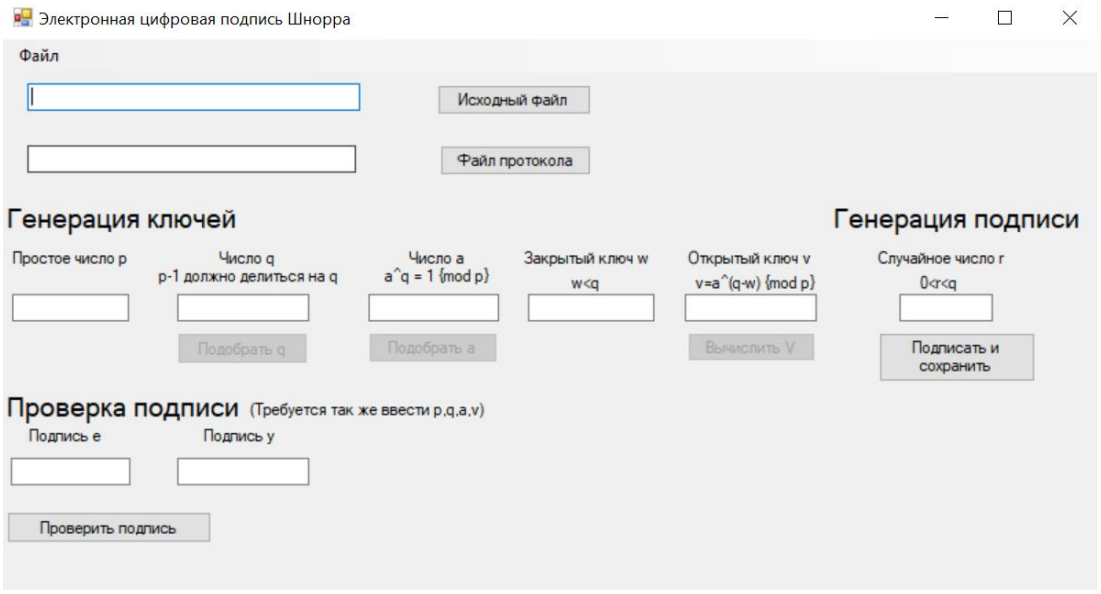

Рис. 1. Окно интерфейса

В левом верхнем углу имеется пункт меню «Файл», предоставляющий пользователю возможность просмотра информации о программе и выхода.

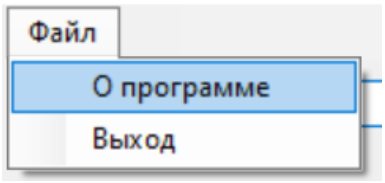

Рис. 2. Пункт меню «Файл»

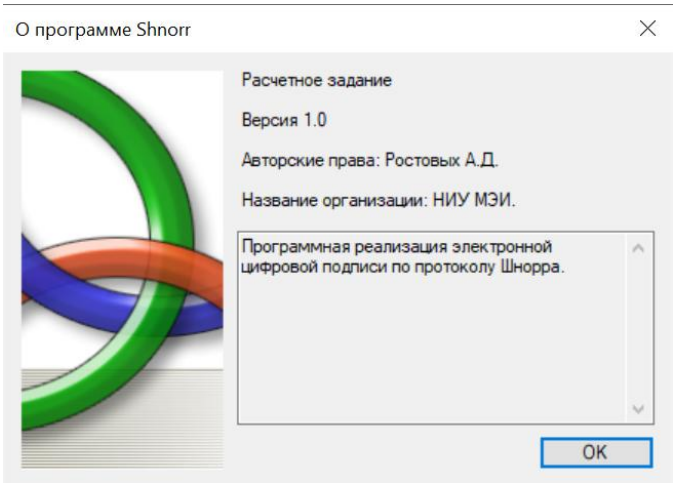

Рис. 3. Окно «О программе»

Для создания подписи пользователю требуется ввести путь к файлу, который требуется подписать и требуемые параметры (числа p,q,a,r, и

открытый и закрытый ключи v и w соответственно), а так же путь к файлу, в котором будут сохранены открытые параметры протокола для дальнейшего просмотра пользователем.

Чтобы избежать сложных вычислений, пользователь может воспользоваться кнопками вычисления требуемых параметров, которые становятся доступными после ввода параметров, требуемых для вычисления или подбора очередного числа (так, например, для подбора q требуется ввести число p (см. рис.4), для вычисления a требуется знать q и p, для v нужно знать a,q,p,w). Ко всем числам даны пояснения к ограничениям.

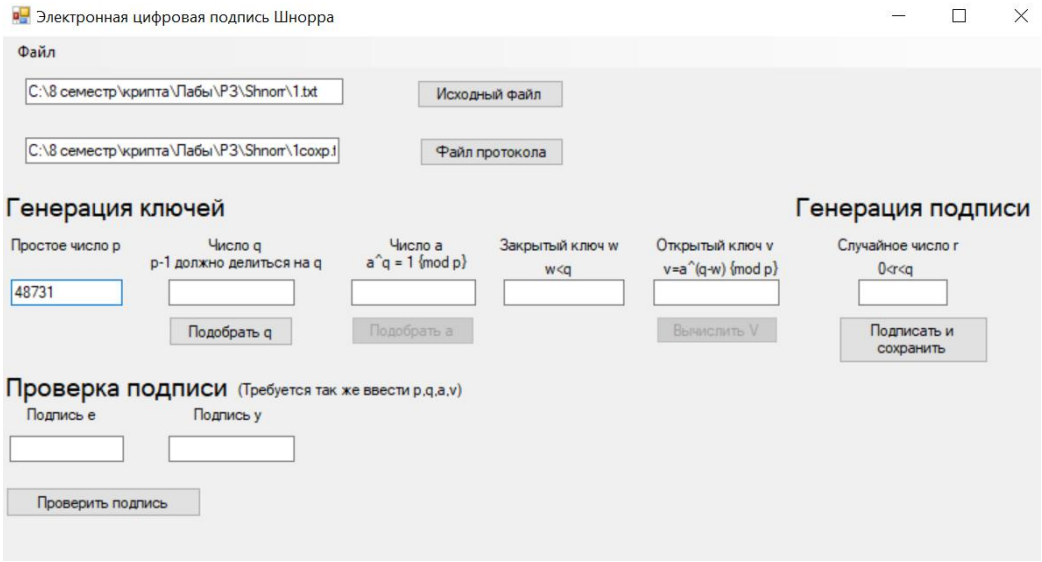

Рис. 4. Пример обновления доступности кнопки «Подобрать q» после ввода числа p.

Если пользователь ввел не все требуемые параметры и пытается создать подпись, появляется соответствующее сообщение:

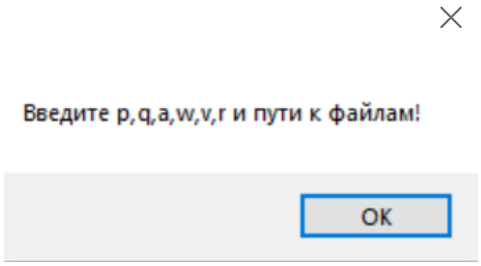

Рис. 5. Сообщение об ошибке.

Если пользователь ввел неверные параметры и пытается создать подпись, пользователю будет выдано сообщение о некорректности параметра:

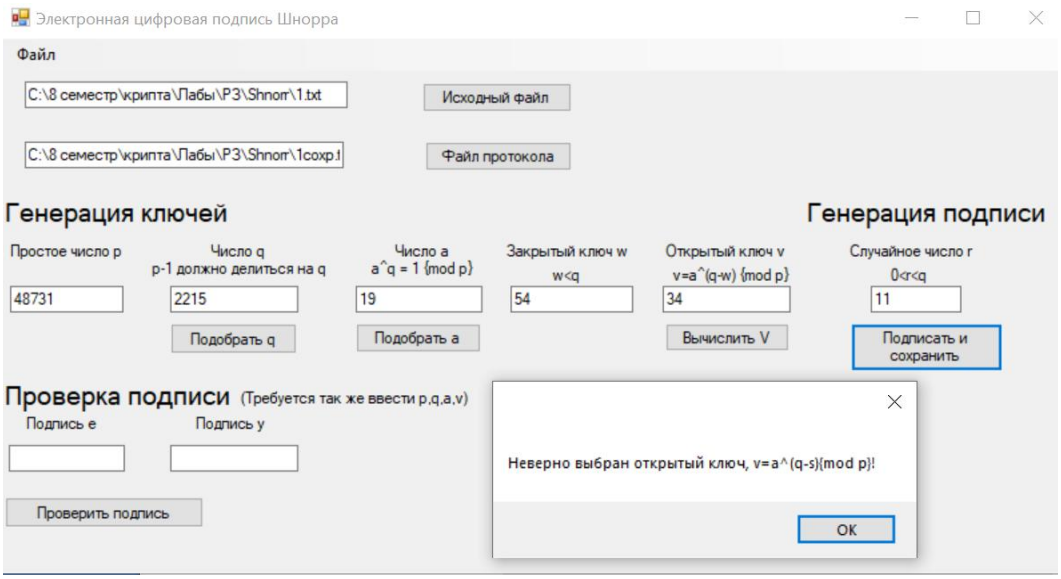

Рис. 6. Сообщение об ошибке в вычислении открытого ключа.

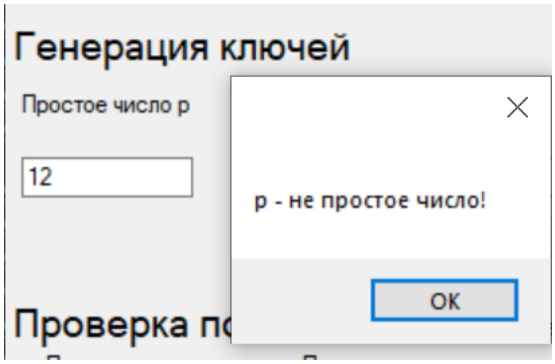

Рис. 7. Сообщение об ошибке ввода простого числа.

При успешном вводе всех параметров и при нажатии кнопки «Создать и сохранить» создается подпись и файл с шагами протокола (рис. 9), который автоматически открывается после закрытия сообщения об успешном создании подписи (рис. 8):

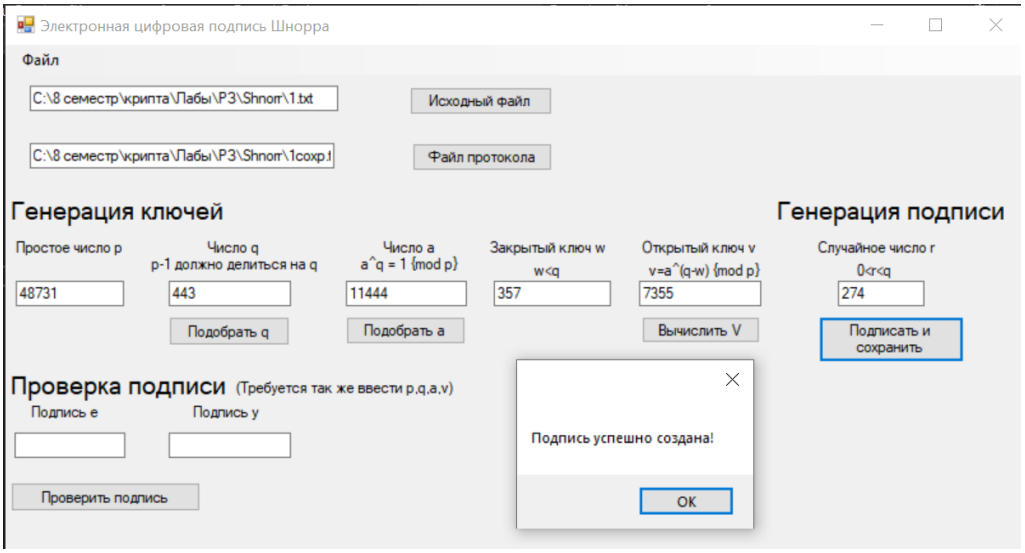

Рис. 8. Пример успешного создания подписи.

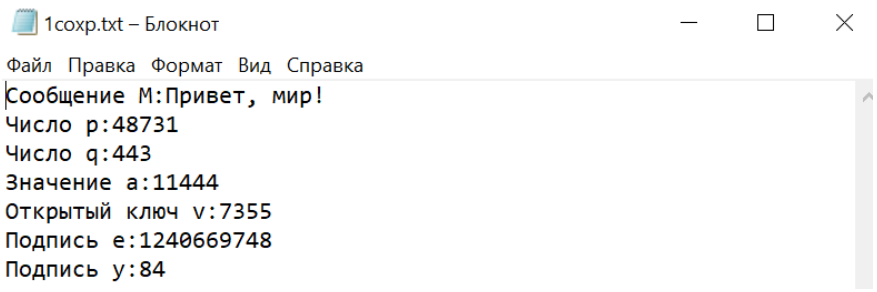

Рис. 9. Информация о шагах протокола.

Чтобы проверить подпись, пользователю требуется ввести две подписи, вычисленные при создании, которые он может узнать из файла или при личной встрече с пользователем, создавшим подпись. Если файл не был искажен и все параметры верны, появится сообщение о подлинности файла:

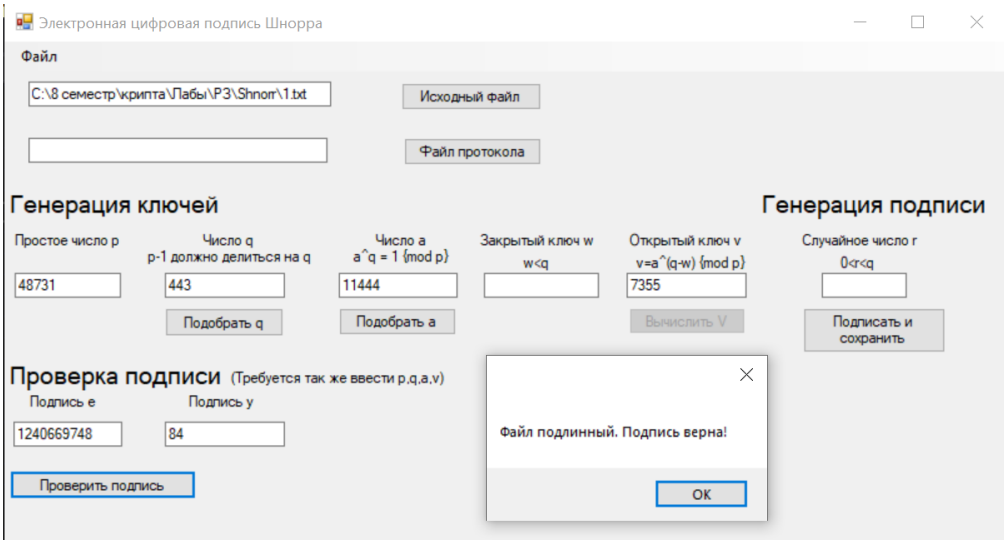

Рис. 10. Пример успешной проверки подписи.

Если же файл был поврежден или искажен (рис. 11), или если пользователь неверно ввел подписи, хэши не совпадут и появится соответствующее сообщение (рис. 12):

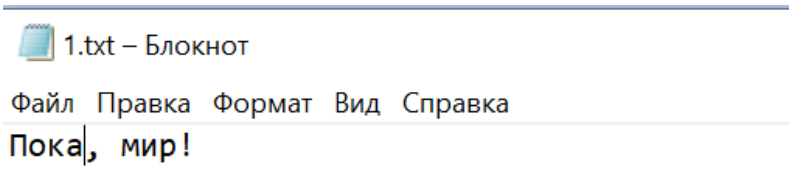

Рис. 11. Искаженный файл.

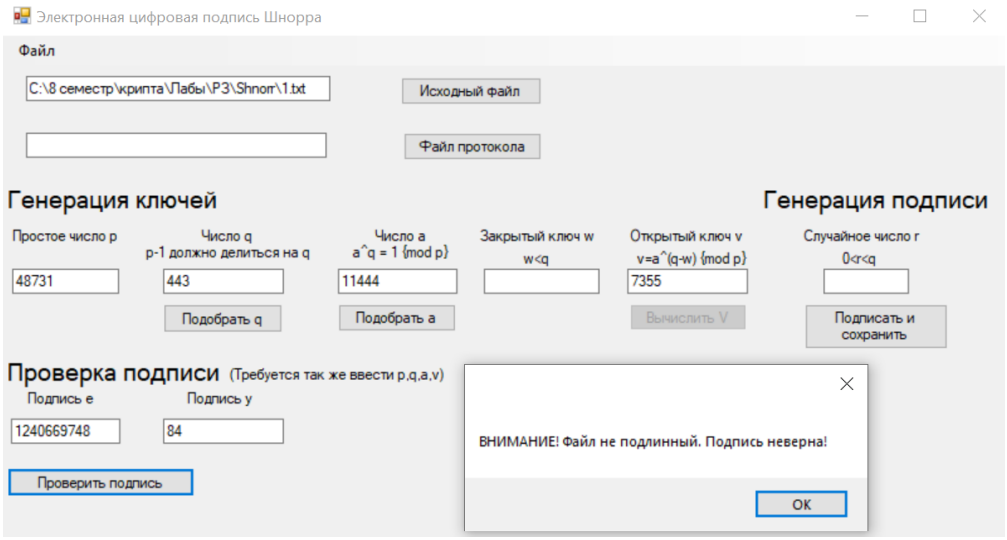

Рис. 12. Сообщение об искаженности файла.

Заключение

Таким образом, в ходе данной работы был рассмотрен один из самых эффективных протоколов создания электронной подписи – протокол Шнорра. Была выполнена реализация данного протокола на языке C#, так же было разработано приложение для создания и проверки электронной подписи по данном протоколу. Была реализована возможность сохранения шагов протокола в файл и последующего просмотра этого файла.

Список использованных источников

1. Схема Шнорра https://ru.wikipedia.org/wiki/%D0%A1%D1%85%D0%B5%D0%BC%D0%B0_%D0%A8%D0%BD%D0%BE%D1%80%D1%80%D0%B0
2. Подпись Шнорра https://translated.turbopages.org/proxy_u/en-ru.ru.d6be2b3b-6277c417-f8dcd690-74722d776562/https/en.wikipedia.org/wiki/Schnorr_signature
3. Цифровая подпись Шнорра https://progler.ru/blog/cifrovaya-podpis-shnorra

Приложение

```
using System;

using System.Collections.Generic;

using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;
using System.IO;

using System.Security.Cryptography;


using System.Numerics;

namespace Shnorr
{
   public partial class Form1 : Form
   {

      public Form1()
      {
         InitializeComponent();
      }


      //проверка: простое ли число?
      private bool IsTheNumberSimple(long n)
      {
         if (n < 2)
            return false;

         if (n == 2)
            return true;

         for (long i = 2; i < n; i++)
```

```
        if (n % i == 0)
            return false;

    return true;

}
 //проверка введенного числа q
private bool IsTheDivider(long p, long q)
{
    if ((p > q) && ((p - 1) % q == 0))
        return true;
    return false;


}
//проверка введенного числа a
private bool IsOKa(long a, long q,long p)
{
    BigInteger aq = BigInteger.Pow(a, (int)q);
    if ((a!=1) && (aq % p == 1))
        return true;
    return false;


}
//проверка введенного закрытого ключа w
private bool IsOKw(long w,long q)
{

    if ((w<q) && (w>0))
        return true;
    return false;


}
//Вычисление числа X
private long Calculate_x(long a, long r, long p)
{

    BigInteger x = BigInteger.Pow(a, (int)r);     //a^r mod p = x
    BigInteger x1 = x % p;
```

```
            return (long)x1;
        }



        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void signbutton_Click(object sender, EventArgs e)
        {
            if ((textBox_p.Text.Length > 0) && (textBox_q.Text.Length > 0) &&
                (textBox_v.Text.Length > 0) && (textBox_a.Text.Length > 0) &&
                (textBox_w.Text.Length > 0) && (textBox_r.Text.Length > 0) &&
                (sourceFilePathTextBox.Text.Length > 0) &&
(signFilePathTextBox.Text.Length > 0))
            {
                long p = Convert.ToInt64(textBox_p.Text);
                long q = Convert.ToInt64(textBox_q.Text);
                long w = Convert.ToInt64(textBox_w.Text);
                long v = Convert.ToInt64(textBox_v.Text);
                long a = Convert.ToInt64(textBox_a.Text);
                long r = Convert.ToInt64(textBox_r.Text);
                if (IsTheNumberSimple(p))
                {
                    if (IsTheDivider(p, q))
                    {
                        if (IsOKa(a,q,p))
                        {
                            if (IsOKw(w, q))
                            {
                                var pkey=BigInteger.Pow(a, (int)(q - w))% p;
                                if (v==pkey)
                                {
                                    if (r<q)
                                    {
                                        long x = Calculate_x(a, r, p);
                                        string str=
File.ReadAllText(sourceFilePathTextBox.Text).ToString();
```

```
                        var xstr= x.ToString();
                        var resstr = string.Concat(str, xstr);
                        string hash = resstr.GetHashCode().ToString();
                        var key = resstr.GetHashCode();
                        if (key < 0)
                        {
                           key *= -1;
                        }
                        var help = r + w * key;
                        long y = help % q;
                        string path = sourceFilePathTextBox.Text;
                        StreamWriter sw = new
StreamWriter(signFilePathTextBox.Text);
                        sw.Write("Сообщение M:"); sw.WriteLine(str);
                        sw.Write("Число p:"); sw.WriteLine(p.ToString());
                        sw.Write("Число q:"); sw.WriteLine(q.ToString());
                        sw.Write("Значение a:"); sw.WriteLine(a.ToString());
                        sw.Write("Открытый ключ v:");
sw.WriteLine(v.ToString());
                        sw.Write("Подпись e:");
sw.WriteLine(key.ToString());
                        sw.Write("Подпись y:"); sw.WriteLine(y.ToString());
                        sw.Close();
                        MessageBox.Show("Подпись успешно создана!");

                        Process.Start(signFilePathTextBox.Text);
                     }
                     else
                        MessageBox.Show("Введите r<q!");
                  }
                  else
                     MessageBox.Show("Неверно выбран открытый ключ,
v=a^(q-s){mod p}!");
               }
               else
                  MessageBox.Show("введите w<q!");
            }
            else
               MessageBox.Show(" Неверно вычислено a, а должно быть
a^q=1{mod p}!");
         }
```

```
                else
                    MessageBox.Show(" q - не является делителем p-1!");
                }
                else
                    MessageBox.Show("p  - не простое число!");
            }
            else
                MessageBox.Show("Введите p,q,a,w,v,r и пути к файлам!");

        }

        //проверка подписи
        private void checkbutton_Click(object sender, EventArgs e)
        {
            if ((textBox_p.Text.Length > 0) && (textBox_q.Text.Length > 0) &&
              (textBox_v.Text.Length > 0) && (textBox_a.Text.Length > 0) &&
              (textBox_e.Text.Length > 0) && (textBox_y.Text.Length > 0) &&
              (sourceFilePathTextBox.Text.Length > 0))
            {
                long p = Convert.ToInt64(textBox_p.Text);
                long q = Convert.ToInt64(textBox_q.Text);
                long pkey = Convert.ToInt64(textBox_e.Text);
                long v = Convert.ToInt64(textBox_v.Text);
                long a = Convert.ToInt64(textBox_a.Text);
                long y = Convert.ToInt64(textBox_y.Text);


                    BigInteger x1 = (BigInteger.ModPow(a, (int)y,(int)p));
                    BigInteger x2 = BigInteger.ModPow(v, (int)pkey,(int)p);
                    BigInteger x = (x1 * x2)%p;
                    //textBox_r.Text= x.ToString();
                    string str =
File.ReadAllText(sourceFilePathTextBox.Text).ToString();
                    var xstr = x.ToString();
                    var resstr = string.Concat(str, xstr);
                    string hash = resstr.GetHashCode().ToString();
                    var key = resstr.GetHashCode();
                    if (key == pkey)
                    {
                        MessageBox.Show("Файл подлинный. Подпись верна!");
```

```
                    }
                    else
                    {
                        MessageBox.Show("ВНИМАНИЕ! Файл не подлинный.
Подпись неверна!");
                    }


                }
                else
                    MessageBox.Show("Введите открытые параметры p,q,a,v,
подписи e и y, и пути к проверяемому файлу!");


            }

            private void srcFile_Click(object sender, EventArgs e)
            {
                OpenFileDialog ofd = new OpenFileDialog();

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

                if (ofd.ShowDialog() == DialogResult.OK)
                {
                    sourceFilePathTextBox.Text = ofd.FileName;
                }
            }

            private void signFile_Click(object sender, EventArgs e)
            {
                OpenFileDialog ofd = new OpenFileDialog();

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

                if (ofd.ShowDialog() == DialogResult.OK)
                {
                    signFilePathTextBox.Text = ofd.FileName;
                }
            }

            private void Rand_Click(object sender, EventArgs e)
```

```
{
    if ((textBox_p.Text.Length > 0))
    {
        long p = Convert.ToInt64(textBox_p.Text);

        if (IsTheNumberSimple(p))
        {


            int[] mass = new int[1000]; //массив для генерируемых чисел
            int j = 0, size_mass = 0;
            for (int i = 2; i < 10000; i++)  //подбор числа q
            {
                if (((p - 1) % i) == 0)
                {
                    mass[j] = i;
                    j++;
                    size_mass++;    //считаем кол-во эл-тов массива
                }
            }
            Random rnd = new Random();
            int qj = rnd.Next(0, size_mass);
            int q = mass[qj];       //если верно, присваиваем значение
            textBox_q.Text = Convert.ToString(q);

            Array.Clear(mass, 0, size_mass);
            size_mass = 0;

        }
        else
            MessageBox.Show("p - не простое число!");
    }
    else
        MessageBox.Show("Введите p!");

}

private void RandAV_Click(object sender, EventArgs e)
{
    if ((textBox_p.Text.Length > 0) && (textBox_q.Text.Length > 0))
    {
```

```
            long p = Convert.ToInt64(textBox_p.Text);

            long q = Convert.ToInt64(textBox_q.Text);
            if (IsTheNumberSimple(p)  && IsTheDivider(p,q))
            {


               int[] mass1 = new int[1000]; //массив для генерируемых чисел
               int j = 0, size_mass1 = 0;

               BigInteger g;   //вводим открытый ключ g

               for (int i = 2; i < 25000; i++)     //поиск g в диапазоне
(15000;25000)
               {

                  g = BigInteger.ModPow(i, (int)q, (int)p);     //i^q mod p = 1
                 if ((g == 1) && (j<1000))     //если выполняется условие
                  {

                     mass1[j] = i;    //заносим в массив

                     j++;
                     size_mass1++;
                  }
               }

              // Console.WriteLine();

              Random rnd1 = new Random();
               int aj = rnd1.Next(0, size_mass1);
               int a = mass1[aj];        //если верно, присваиваем значение
               textBox_a.Text = Convert.ToString(a);

               Array.Clear(mass1, 0, size_mass1);     //очищаем массив
               size_mass1 = 0;      //обнуляем кол-во элементов в массиве

            }
            else
               MessageBox.Show("Проверьте правильность введенных чисел!
p-простое число, p-1 делится на q");
```

```
            }
            else
                MessageBox.Show("Введите p, q!");

        }

        private void CalcV_Click(object sender, EventArgs e)
        {
            if ((textBox_p.Text.Length > 0) && (textBox_w.Text.Length > 0) &&
(textBox_q.Text.Length > 0))
            {
                long p = Convert.ToInt64(textBox_p.Text);
                long w = Convert.ToInt64(textBox_w.Text);
                long q = Convert.ToInt64(textBox_q.Text);
                long a = Convert.ToInt64(textBox_a.Text);
                if (IsTheNumberSimple(p) && IsOKw(w, q) && IsTheDivider(p, q)
&& IsOKa(a,q,p))
                {

                    var pkey = BigInteger.Pow(a, (int)(q - w)) % p;

                   textBox_v.Text = Convert.ToString(pkey);
                }
                else
                    MessageBox.Show("Проверьте правильность введенных чисел!
p-простое число, p-1 делится на q, w меньше q, a^q=1 {mod p}");
            }
            else
                MessageBox.Show("Введите p, q , число a и  закрытый ключ w!");
        }

        private void textBox_p_TextChanged(object sender, EventArgs e)
        {

        }

        private void textBox_q_TextChanged(object sender, EventArgs e)
        {

        }
```

```
        private void textBox_TextChanged(object sender, EventArgs e)
        {
            RandAV.Enabled = (textBox_p.Text.Length > 0) &&
(textBox_q.Text.Length > 0);
            Rand.Enabled = textBox_p.Text.Length > 0;
            CalcV.Enabled = (textBox_p.Text.Length > 0) &&
(textBox_q.Text.Length > 0) &&
              (textBox_a.Text.Length > 0) && (textBox_w.Text.Length > 0);
        }

        private void оПрограммеToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            AboutBox1 AB1 = new AboutBox1();
            AB1.labelProductName.Text = "Расчетное задание";
            AB1.labelVersion.Text = "Версия 1.0";
            AB1.labelCopyright.Text = "Авторские права: Ростовых А.Д.";
            AB1.labelCompanyName.Text = "Название организации: НИУ
МЭИ.";
            AB1.textBoxDescription.Text = "Программная реализация
электронной цифровой подписи по протоколу Шнорра.";
            AB1.ShowDialog();
        }

        private void выходToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```