

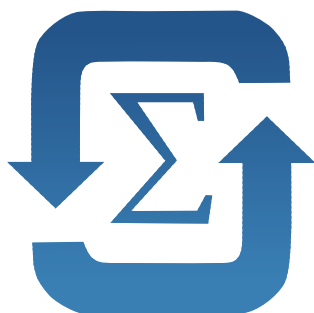
Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное
учреждение высшего образования
«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Кафедра теоретической и прикладной информатики

Лабораторная работа № 3
по дисциплине «Информационная безопасность»



ФАКУЛЬТЕТ:	ПМИ
ГРУППА:	ПМИ-61
СТУДЕНТЫ:	Ершов П.К., Мамонова Е.В., Цыденов З.Б.
ВАРИАНТ:	2
ПРЕПОДАВАТЕЛЬ:	Авдеенко Т.В.

Новосибирск

2020

1. Цель работы

Ознакомиться с существующими криптографическими библиотеками. Научиться использовать сторонние криптографические библиотеки при разработке собственных приложений.

2. Задание

I. Найти криптографическую библиотеку.

II. Реализовать приложение с графическим интерфейсом, позволяющее выполнять следующие действия.

1. Шифровать и дешифровать выбранный пользователем файл с использованием одного или нескольких симметричных алгоритмов шифрования:
 - 1) результаты шифрования и дешифрования должны сохраняться в файлы;
 - 2) требуемые параметры шифрования, такие как ключ, вектор инициализации и пр., должны считываться из файла.
2. Шифровать и дешифровать выбранный пользователем файл с использованием одного асимметричного алгоритма шифрования:
 - 1) реализовать процедуру генерации пары открытый–закрытый ключ;
 - 2) результаты шифрования и дешифрования должны сохраняться в файлы;
 - 3) требуемые параметры шифрования должны считываться из файла.
3. Вычислять и проверять электронную цифровую подпись для выбранного пользователем файла с использованием одного из алгоритмов
 - 1) результаты вычисления подписи должны сохраняться в файлы;
 - 2) требуемые параметры вычисления и проверки подписи должны считываться из файла.
4. Вычислять хэш-значения для выбранного пользователем файла по одному или нескольким алгоритмам хэширования, при этом вычисленное хэш-значение должно сохраняться в файл.

III. Протестировать правильность работы реализованного приложения.

3. Описание разработанного программного средства

Была выбрана криптографическая библиотека языка C# System.Security.Cryptography.

В качестве демонстрационных алгоритмов были выбраны:

Для симметричных алгоритмов шифрования:

1. 3DES.
2. AES.

Для асимметричных алгоритмов шифрования:

1. RSA.

Для создания цифровых подписей:

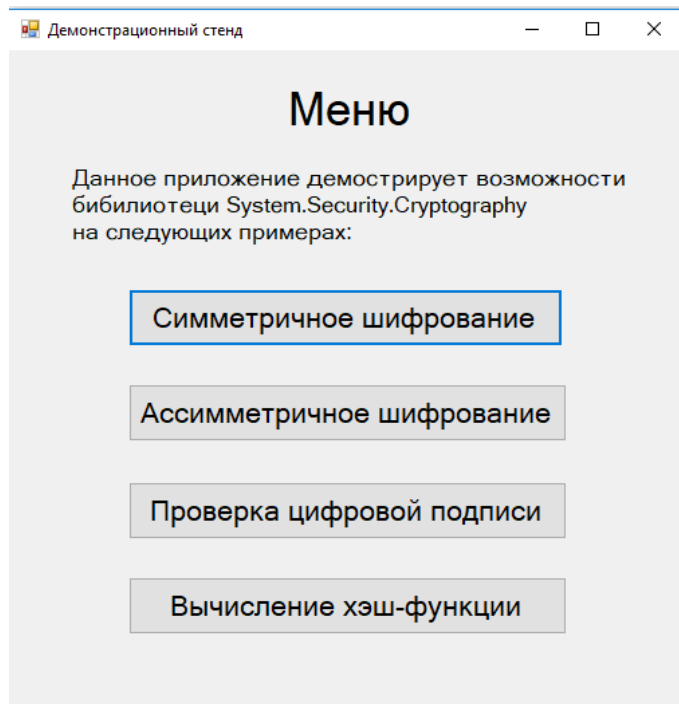
1. DSA.
2. ECDSA.
3. HMACSHA512 (алгоритм HMAC на основе SHA512).

Для создания хэшей:

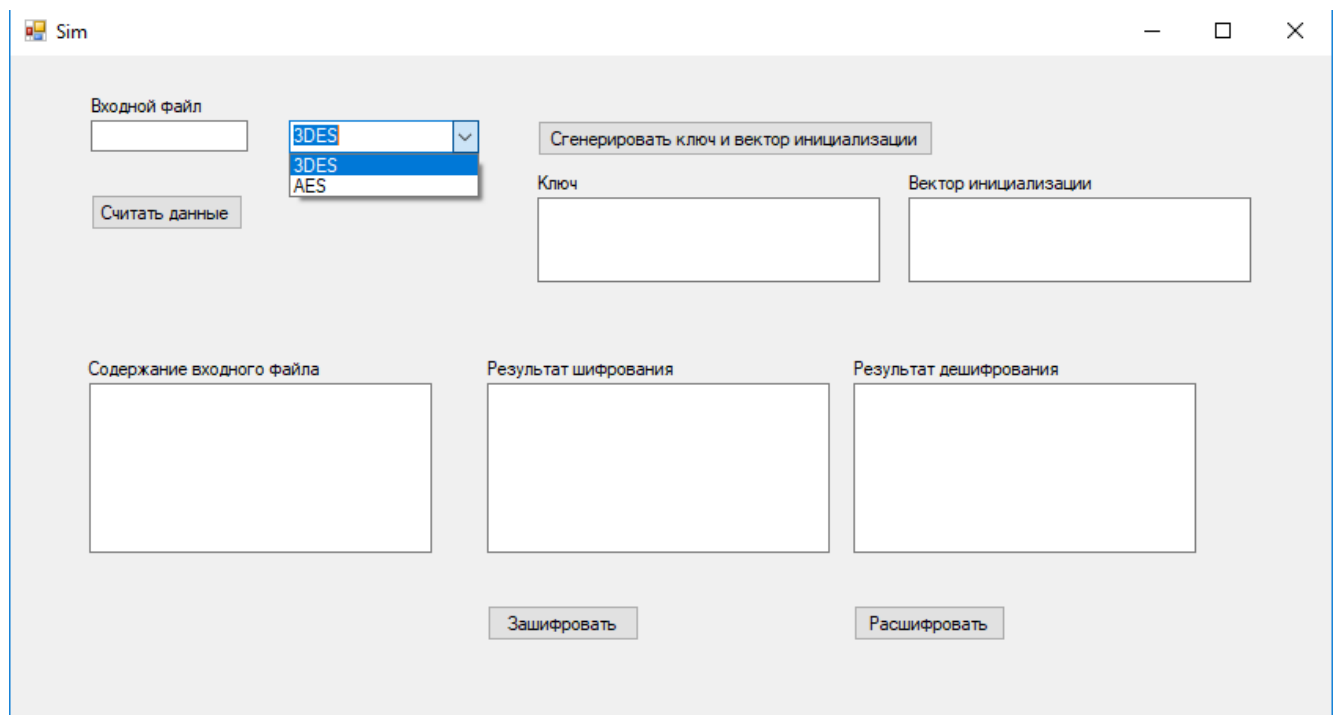
1. SHA512.
2. RIPEMD160.

Разработанное программное средство способно считывать необходимые данные из файлов и записывать результаты в выходные файлы. Так же разработанное приложение способно учитывать некорректные данные.

Основное меню:



Симметричное шифрование:



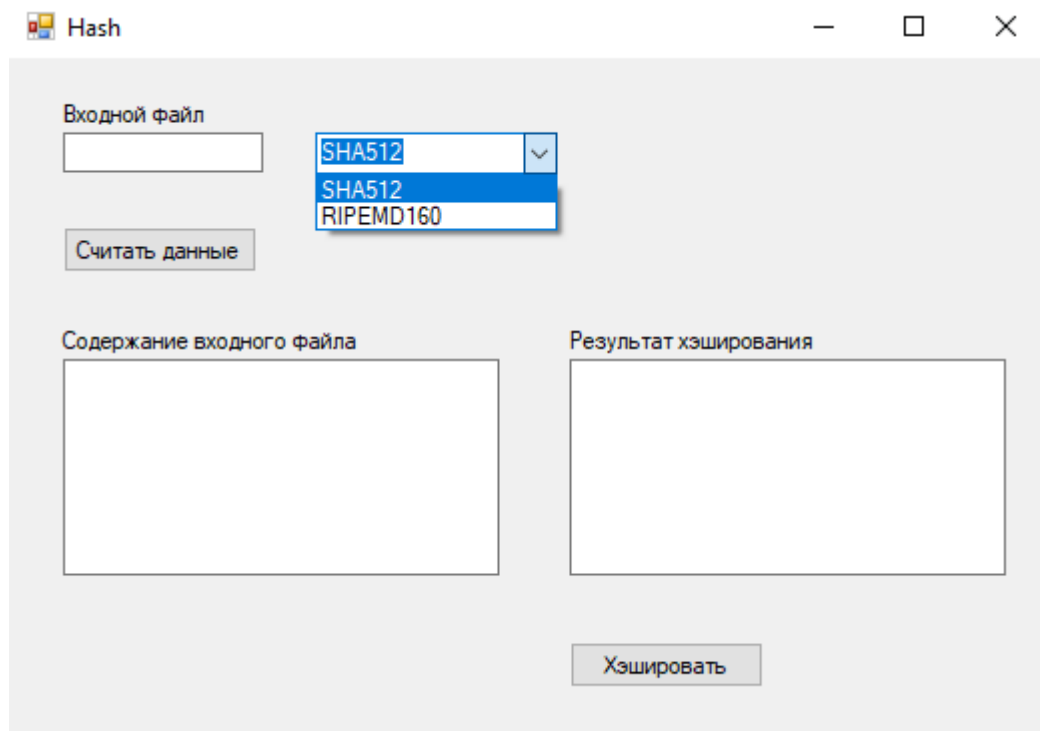
Асимметричное шифрование:

The Assim application window is titled "Assim" and features a standard Windows title bar with minimize, maximize, and close buttons. The interface is designed for demonstrating asymmetric encryption using the RSA algorithm. At the top, a text label states "Асимметричное шифрование демонстрируется на примере RSA алгоритма". Below this, there is a "Входной файл" (Input file) section with a text input field and a "Считать данные" (Load data) button. The main area is divided into three columns: "Содержание входного файла" (Input file content) with a large text area, "Результат шифрования" (Encryption result) with a large text area, and "Результат дешифрования" (Decryption result) with a large text area. At the bottom, there are two buttons: "Зашифровать" (Encrypt) and "Расшифровать" (Decrypt).

Цифровые подписи:

The Dig_key application window is titled "Dig_key" and features a standard Windows title bar with minimize, maximize, and close buttons. The interface is designed for generating and verifying digital signatures. It includes a "Входной файл" (Input file) section with a text input field, a "Считать данные" (Load data) button, and a dropdown menu for selecting a signature algorithm (DSA, ECDSA, HMACSHA512). A "Сгенерировать ключи" (Generate keys) button is located to the right. The main area is divided into two columns: "Содержание входного файла" (Input file content) with a large text area, and "Цифровая подпись" (Digital signature) with a large text area. To the right of the signature area, there are fields for "Количество бит в подписи" (Number of bits in signature) and "Номер бита подписи" (Signature bit number), along with a "Модифицировать подпись" (Modify signature) button. At the bottom, there are buttons for "Сгенерировать подпись" (Generate signature) and "Проверка подписи" (Verify signature).

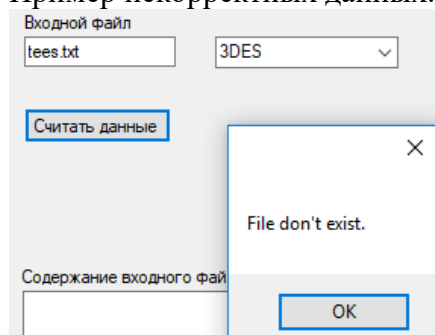
Хэш-функции:



4. Исследования

4.1. Симметричное шифрование.

Пример некорректных данных:



Демонстрация 3DES:

Sim

Входной файл

test.txt

3DES

Сгенерировать ключ и вектор инициализации

Считать данные

Ключ

c9 2d 3a db c3 ec 48 1a af b1 9b 26 96 6f 99 cf

Вектор инициализации

d5 e4 14 95 b0 fe 10 75 92 f8 f5 34 c9 8b e7 a9

Содержание входного файла

f12eref d12 x12 dasvf qwfqawcqw fqew ff
aihfiqwy fqiwy fpiqiyyq 7igewufyg

Результат шифрования

97 a2 5e f8 ee bb f6 88 d4 c ed d f9 17 e9
9b 64 59 c9 46 8d 57 71 11 49 f2 8b 8b 20
69 fc 62 a2 d9 25 e5 7 31 c1 93 51 f1 1a 65
7d 5c 8a 30 0 cb 92 1d a1 11 ae 95 f5 5f f3
9a b2 2c 10 fe d5 4c fb 1 c3 8f e5 fb e 6b
96 da 90 86 15 6d

Результат дешифрования

f12eref d12 x12 dasvf qwfqawcqw fqew ff
aihfiqwy fqiwy fpiqiyyq 7igewufyg

Зашифровать

Расшифровать

Демонстрация AES:

Sim

Входной файл

test.txt

AES

Сгенерировать ключ и вектор инициализации

Считать данные

Ключ

ae 46 47 85 c 60 40 e7 b1 80 9a d1 52 b4 d4 c0

Вектор инициализации

60 27 ad b9 a9 d0 c4 5f8 32 8a 6f 7b 41 14 16

Содержание входного файла

f12eref d12 x12 dasvf qwfqawcqw fqew ff
aihfiqwy fqiwy fpiqiyyq 7igewufyg

Результат шифрования

d2 aa cf 66 fa 75 e1 1 16 ed 50 15 96 be e9
cc 74 5e 3d aa 2d 7a 29 e0 7a a6 50 4a 1e
10 cd 6b 2f d7 c 22 13 52 63 0 b6 1e c1 97
e3 4a 98 b0 8f c8 d7 a5 f e7 ad 2e 7b 83 b8
7 a6 43 d2 55 b1 e3 ff 21 2c 19 93 c0 1e ea
4b 6b 86 16 d3 97

Результат дешифрования

f12eref d12 x12 dasvf qwfqawcqw fqew ff
aihfiqwy fqiwy fpiqiyyq 7igewufyg

Зашифровать

Расшифровать

4.2. Асимметричное шифрование.

Демонстрация RSA:

Assim

Входной файл
test.txt

Считать данные

Асимметричное шифрование демонстрируется на примере RSA алгоритма

Содержание входного файла
f12eref d12 x12 dasvf qwfqawcqew fqew ff
aihfiqwy fqjwy fpiqiyq 7lgewufyg

Результат шифрования
a0 99 22 79 dd 65 ff 90 6d 73 ce 5b 14 cb
4f b8 78 67 ab 87 d8 73 6f 9d 48 9b df 44
68 29 ec ab d4 d5 a1 18 7d 77 c3 d4 88 e1
53 f6 6c 8e aa fd 60 a2 7d 8 44 12 44 c6 e5
57 37 c7 28 4d 1e 73 2a 43 a5 b cd 9f 83
99 2a 7b cc 97 d3 42 29 aa 9d c2 e4 4c eb
6a 1f 5c f1 6d 5a 3a 3 8 5f6 91 9f d3 b4 b8

Результат дешифрования
f12eref d12 x12 dasvf qwfqawcqew fqew ff
aihfiqwy fqjwy fpiqiyq 7lgewufyg

Зашифровать

Расшифровать

4.3. Цифровые подписи.

Демонстрация DSA с модификацией подписи:

Dig_key

Входной файл
test.txt

DSA

Сгенерировать ключи

Считать данные

Содержание входного файла
f12eref d12 x12 dasvf qwfqawcqew fqew ff
aihfiqwy fqjwy fpiqiyq 7lgewufyg

Цифровая подпись
7c 42 33 96 fb 22 e5 e4 1b 69 91 cc 5d c2
68 b0 53 5d 28 7f 33 7b b7 43 70 3e 76 a1
c4 88 60 6f 4e 13 10 80 a3 d6 46 4c 5b d7
45 d8 3b cf 61 ad ba eb 6d 3b 75 4b 21 62
8a 39 ea d8 34 c1 0 d1

Количество бит в подписи
512

Номер бита подписи

Модифицировать подпись

Сгенерировать подпись

Результат проверки подписи
Верный

Проверка подписи

Dig_key

Входной файл
test.txt

DSA

Сгенерировать ключи

Считать данные

Содержание входного файла
f12eref d12 x12 dasvf qwfqawcqew fqew ff
aihfiqwy fqjwy fpiqiyq 7lgewufyg

Цифровая подпись
7c 42 33 96 fb 22 e5 e4 1b 69 91 cc 5d c2
68 b0 53 5d 28 7f 33 7b b7 43 70 3e 76 a1
c4 88 60 6f 4e 13 10 80 a3 d6 46 4c 5b d7
45 d8 3b cf 61 ad ba eb 6d 3b 75 4b 21 62
8a 39 ea d8 34 c1 0 d1

Количество бит в подписи
512

Номер бита подписи
12

Модифицировать подпись

Сгенерировать подпись

Результат проверки подписи
Не верный

Проверка подписи

Демонстрация ECDSA с модификацией подписи:

Входной файл

test.txt

ECDSA

Сгенерировать ключи

Считать данные

Содержание входного файла

f12eref d12 x12 dasvf qwfqawcqw fqew ff
aihfiqwy fqiwypriqiyuq7igewufyg

Цифровая подпись

0 7e fa 84 a8 7a 9a 15 26 8c 98 9b 55 2f 22
a0 17 3 46 b5 f 2b 5e 84 87 fd 75 34 ed c6
2e d1 ae f 3d 1e cb 5 9f f1 68 53 4f 12 60
39 da 5c 6d f5 2a e7 8c 7f 4f a0 2 43 3b a3
95 7d 6 8a dc 1e 0 85 d0 fe a6 be 50 89 7b
9f 6a e4 59 73 78 f8 5c 4e 4a c0 79 b6 17
99 72 ee 15 5c ff f0 39 41 3d 1f 55 c9 4f a7

Количество бит в подписи

1056

Номер бита подписи

Модифицировать подпись

Сгенерировать подпись

Результат проверки подписи

Верный

Проверка подписи

Входной файл

test.txt

ECDSA

Сгенерировать ключи

Считать данные

Содержание входного файла

f12eref d12 x12 dasvf qwfqawcqw fqew ff
aihfiqwy fqiwypriqiyuq7igewufyg

Цифровая подпись

0 7e fa 84 a8 7a 9a 15 26 8c 98 9b 55 2f 22
a0 17 3 46 b5 f 2b 5e 84 87 fd 75 34 ed c6
2e d1 ae f 3d 1e cb 5 9f f1 68 53 4f 12 60
39 da 5c 6d f5 2a e7 8c 7f 4f a0 2 43 3b a3
95 7d 6 8a dc 1e 0 85 d0 fe a6 be 50 89 7b
9f 6a e4 59 73 78 f8 5c 4e 4a c0 79 b6 17
99 72 ee 15 5c ff f0 39 41 3d 1f 55 c9 4f a7

Количество бит в подписи

1056

Номер бита подписи

120

Модифицировать подпись

Сгенерировать подпись

Результат проверки подписи

Не верный

Проверка подписи

Демонстрация HMACSHA512 с модификацией подписи:

Входной файл

test.txt

HMACSHA512

Сгенерировать ключи

Считать данные

Содержание входного файла

f12eref d12 x12 dasvf qwfqawcqw fqew ff
aihfiqwy fqiwypriqiyuq7igewufyg

Цифровая подпись

40 2b 20 30 bb c9 fe c0 53 32 6e ef 2f a8
3d 15 4f dc b8 26 cb 71 ba 36 82 27 3c f9
3a b6 f5 97 41 18 a8 83 58 10 c2 95 87 e3
d1 5a 21 e6 c3 4d 91 1c 66 b1 2f fb ae 3 66
d1 a 6a c5 6f 2 e4

Количество бит в подписи

512

Номер бита подписи

Модифицировать подпись

Сгенерировать подпись

Результат проверки подписи

Верный

Проверка подписи

Входной файл

test.txt

HMACSHA512

Сгенерировать ключи

Считать данные

Содержание входного файла

f12eref d12 x12 dasvf qwfqawcqw fqew ff
aihfiqwy fqiwypriqiyuq7igewufyg

Цифровая подпись

40 2b 20 30 bb c9 fe c0 53 32 6e ef 2f a8
3d 15 4f dc b8 26 cb 71 ba 36 82 27 3c f9
3a b6 f5 97 41 18 a8 83 58 10 c2 95 87 e3
d1 5a 21 e6 c3 4d 91 1c 66 b1 2f fb ae 3 66
d1 a 6a c5 6f 2 e4

Количество бит в подписи

512

Номер бита подписи

430

Модифицировать подпись

Сгенерировать подпись

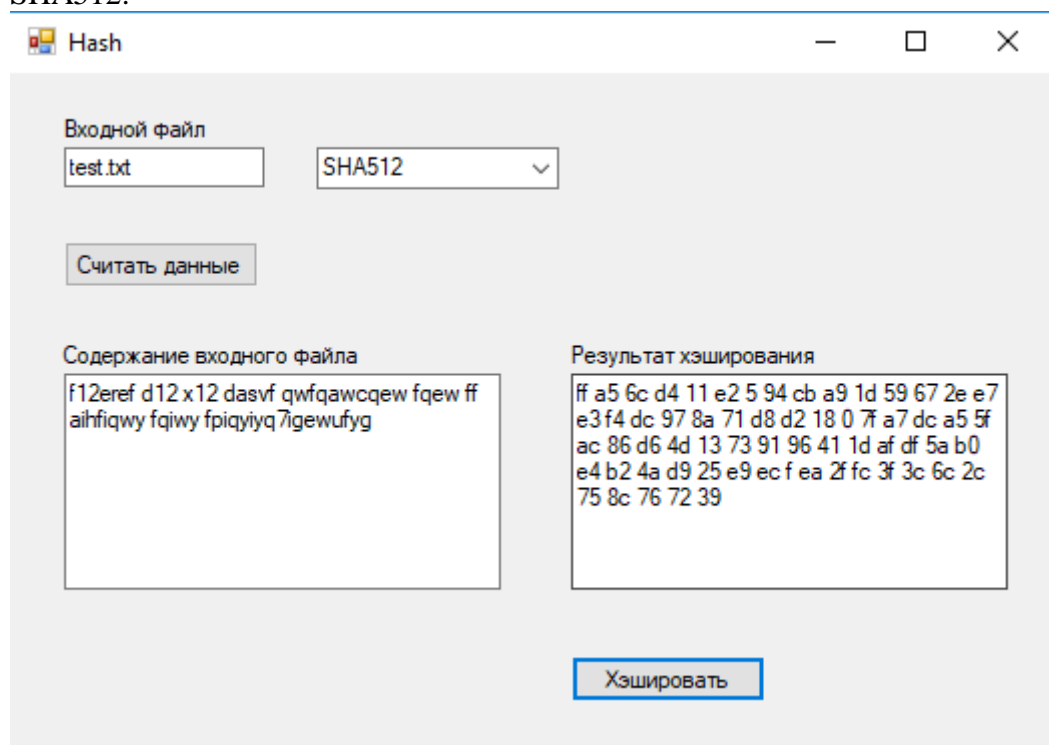
Результат проверки подписи

Не верный

Проверка подписи

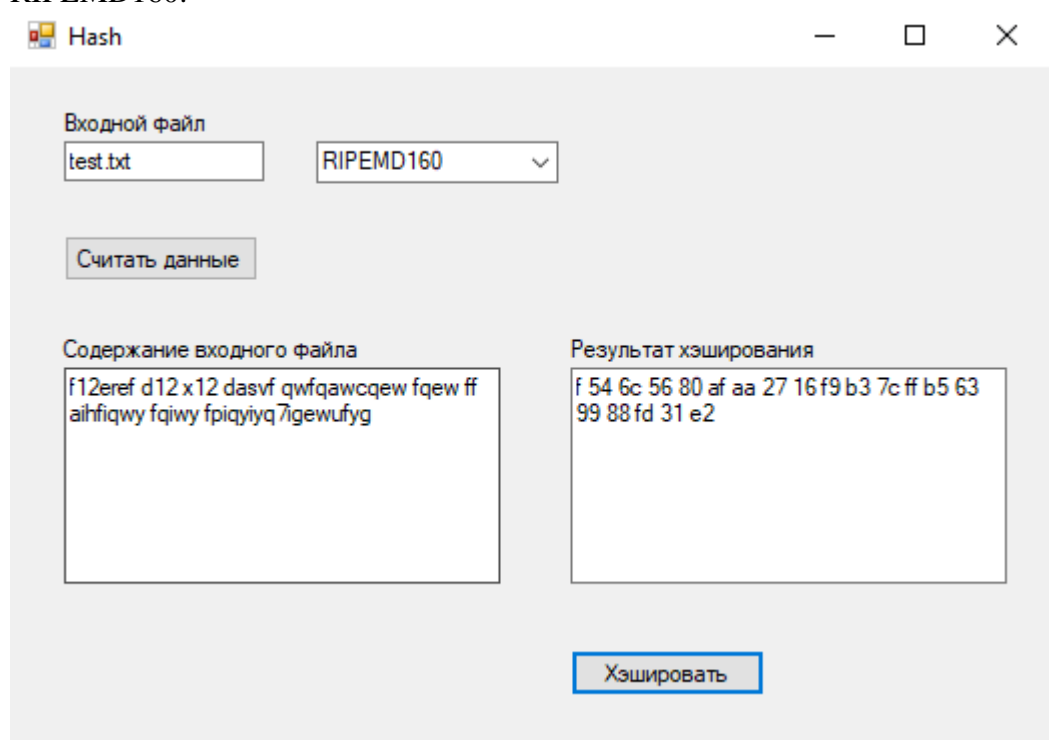
4.4. Хэш-функции.

SHA512:



The screenshot shows a window titled "Hash" with a standard Windows interface (minimize, maximize, close buttons). Inside the window, there are two input fields at the top: "Входной файл" (Input file) containing "test.txt" and a dropdown menu set to "SHA512". Below these is a button labeled "Считать данные" (Load data). The main area is divided into two text boxes. The left box, labeled "Содержание входного файла" (Content of input file), contains the text: "f12eref d12 x12 dasvf qwfqawcqew fqew ff aihfiqwy fqjwy fpiqiyyq 7igewufyg". The right box, labeled "Результат хэширования" (Hashing result), displays the SHA512 hash: "ff a5 6c d4 11 e2 5 94 cb a9 1d 59 67 2e e7 e3 f4 dc 97 8a 71 d8 d2 18 0 7f a7 dc a5 5f ac 86 d6 4d 13 73 91 96 41 1d af df 5a b0 e4 b2 4a d9 25 e9 ec f ea 2f fc 3f 3c 6c 2c 75 8c 76 72 39". At the bottom right, there is a button labeled "Хэшировать" (Hash).

RIPEMD160:



The screenshot shows the same "Hash" application window, but with the dropdown menu set to "RIPEMD160". The "Входной файл" (Input file) field still contains "test.txt", and the "Считать данные" (Load data) button is present. The "Содержание входного файла" (Content of input file) text box remains the same. The "Результат хэширования" (Hashing result) text box now displays the RIPEMD160 hash: "f 54 6c 56 80 af aa 27 16 f9 b3 7c ff b5 63 99 88 fd 31 e2". The "Хэшировать" (Hash) button is still at the bottom right.

5. Код программы

Sym_alg.cs

```
class Sym_alg
{
    public byte[] key_gen() // генератор 128-битного ключа
    {
        byte[] key = new byte[128]; // основной ключ
        byte[] b1 = new byte[64]; // 64-битная часть ключа
        byte[] b2 = new byte[64]; // 64-битная часть ключа
        byte[] xor = new byte[64]; // 64-битная блок случайных бит для равномерного преобразования ключа
        byte[] bit = new byte[1]; // случайный бит для выбора совмещения двух частей ключа (1 - !b1 + b2, 0 - b2 + !b1)

        Thread.Sleep(20); // тормозим генерацию ключа, чтобы новый ключ был случайным
        var rand = new Random((int)DateTime.Now.Ticks & 0x0000FFFF);
        rand.NextBytes(b1); // создаём случайную половину ключа
        rand.NextBytes(b2); // создаём случайную половину ключа

        rand.NextBytes(xor); // создаём случайную последовательность для операции хог с половинами ключа
        ulong k1 = BitConverter.ToUInt64(b1, 0); // переводим в ulong формат
        ulong k2 = BitConverter.ToUInt64(b2, 0); // переводим в ulong формат

        ulong xr = BitConverter.ToUInt64(xor, 0); // переводим в ulong формат
        k1 ^= xr; // приводим к равномерному виду
        xr = BitConverter.ToUInt64(xor, 0);
        k2 ^= xr; // приводим к равномерному виду

        rand.NextBytes(bit); // создаём случайное число (0 или 1), чтобы выбрать, какую половину ключа поставить первой и
        // какую инвертировать
        if (bit[0] == 1)
        {
            b1 = BitConverter.GetBytes(~k1); // обращаем число
            b2 = BitConverter.GetBytes(k2);
            key = b1.Concat(b2).ToArray(); // соединяем половины 128-битного ключа
        }
        else
        {
            b1 = BitConverter.GetBytes(k1);
            b2 = BitConverter.GetBytes(~k2); // обращаем число
            key = b2.Concat(b1).ToArray(); // соединяем половины 128-битного ключа
        }

        return key;
    }
    public byte[] TriDES_Encrypt(byte[] Data, byte[] Key, byte[] IV) // 3DES шифратор
    {
        MemoryStream mStream = new MemoryStream(); // создаём поток памяти

        TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider(); // объект класса 3DES
        //tdes.Padding = PaddingMode.None; //отключаем дополнение выходного шифротекста справа
        tdes.Mode = CipherMode.CFB; // включаем режим CFB, чтобы установить длину шифротекста в 64 бита

        CryptoStream cStream = new CryptoStream(mStream,
            tdes.CreateEncryptor(Key, IV),
            CryptoStreamMode.Write); // создаём поток для шифрования

        byte[] toEncrypt = Data;

        cStream.Write(toEncrypt, 0, toEncrypt.Length); // помещаем данные в поток шифрования
        cStream.FlushFinalBlock();

        byte[] ret = mStream.ToArray(); // извлекаем шифротекст

        cStream.Close(); // закрываем потоки
        mStream.Close();

        return ret;
    }
    public byte[] TriDES_Decrypt(byte[] Data, byte[] Key, byte[] IV) // 3DES дешифратор
    {
        MemoryStream mStream = new MemoryStream(); // создаём поток памяти

        TripleDESCryptoServiceProvider tdes = new TripleDESCryptoServiceProvider(); // объект класса 3DES
        //tdes.Padding = PaddingMode.None; //отключаем дополнение выходного шифротекста справа
        tdes.Mode = CipherMode.CFB; // включаем режим CFB, чтобы установить длину шифротекста в 64 бита

        CryptoStream cStream = new CryptoStream(mStream,
            tdes.CreateDecryptor(Key, IV),
            CryptoStreamMode.Write); // создаём поток для дешифрования

        byte[] toEncrypt = Data;

        cStream.Write(toEncrypt, 0, toEncrypt.Length); // помещаем данные в поток шифрования
        cStream.FlushFinalBlock();
    }
}
```

```

        byte[] ret = mStream.ToArray(); // извлекаем текст

        cStream.Close(); // закрываем потоки
        mStream.Close();

        return ret;
    }

    public byte[] AES_Encrypt(byte[] Data, byte[] Key, byte[] IV) // AES шифратор
    {
        MemoryStream mStream = new MemoryStream(); // создаём поток памяти

        Aes AES = Aes.Create(); // объект класса AES
        AES.Mode = CipherMode.CFB; // включаем режим CFB, чтобы установить длину шифротекста в 64 бита

        CryptoStream cStream = new CryptoStream(mStream,
            AES.CreateEncryptor(Key, IV),
            CryptoStreamMode.Write); // создаём поток для шифрования

        byte[] toEncrypt = Data;

        cStream.Write(toEncrypt, 0, toEncrypt.Length); // помещаем данные в поток шифрования
        cStream.FlushFinalBlock();

        byte[] ret = mStream.ToArray(); // извлекаем шифротекст

        cStream.Close(); // закрываем потоки
        mStream.Close();

        return ret;
    }

    public byte[] AES_Decrypt(byte[] Data, byte[] Key, byte[] IV) // AES дешифратор
    {
        MemoryStream mStream = new MemoryStream(); // создаём поток памяти

        Aes AES = Aes.Create(); // объект класса 3DES
        AES.Mode = CipherMode.CFB; // включаем режим CFB, чтобы установить длину шифротекста в 64 бита

        CryptoStream cStream = new CryptoStream(mStream,
            AES.CreateDecryptor(Key, IV),
            CryptoStreamMode.Write); // создаём поток для дешифрования

        byte[] toEncrypt = Data;

        cStream.Write(toEncrypt, 0, toEncrypt.Length); // помещаем данные в поток дешифрования
        cStream.FlushFinalBlock();

        byte[] ret = mStream.ToArray(); // извлекаем текст

        cStream.Close(); // закрываем потоки
        mStream.Close();

        return ret;
    }
}

```

Sym.cs

```

public partial class Sym : Form
{
    public Sym()
    {
        InitializeComponent();
        comboBox1.Items.Add("3DES");
        comboBox1.Items.Add("AES");
        comboBox1.SelectedIndex = 0;
    }

    byte[] b_mess = new byte[1]; // сообщение в байтах
    string[] mess = new string[1]; // сообщение в строках
    string alg_flag = ""; // название алгоритма
    byte[] encrypt = new byte[1]; // шифротекст
    byte[] decrypt = new byte[1]; // дешифрованное сообщение
    byte[] KEY = new byte[1]; // ключ
    byte[] IV = new byte[1]; // вектор инициализации

    private void Button3_Click(object sender, EventArgs e)
    {
        File_f F = new File_f();
        string file_name = textBox1.Text;
        if (file_name == "") // проверка на ввод имени файла
            MessageBox.Show("Enter file name.");
        else
        {

```

```

        bool fl = File.Exists(file_name); // проверка на существование файла
        if (fl)
        {
            b_mess = File.ReadAllBytes(file_name);
            mess = F.Read(file_name);
            string ms = "";
            for (int i = 0; i < mess.Length; i++)
                ms += mess[i];
            textBox2.Text = ms;
        }
        else
            MessageBox.Show("File don't exist.");
    }
}

private void Button1_Click(object sender, EventArgs e) // шифруем сообщения
{
    textBox3.Clear();

    string file_name = "";
    alg_flag = comboBox1.SelectedItem.ToString();
    Sym_alg SM = new Sym_alg();

    KEY = File.ReadAllBytes("key.bin");
    IV = File.ReadAllBytes("iv.bin");

    if (alg_flag == "3DES")
    {
        encrypt = SM.TriDES_Encrypt(b_mess, KEY, IV);
        file_name = "enc_3DES.bin";
    }
    if (alg_flag == "AES")
    {
        encrypt = SM.AES_Encrypt(b_mess, KEY, IV);
        file_name = "enc_AES.bin";
    }

    File.WriteAllBytes(file_name, encrypt);

    for (int i = 0; i < encrypt.Length; i++)
        textBox3.Text += encrypt[i].ToString("x") + " ";
}

private void Button2_Click(object sender, EventArgs e) // дешируем сообщения
{
    textBox4.Clear();

    Sym_alg SM = new Sym_alg();
    string file_name = "";
    if (alg_flag == "3DES")
    {
        decrypt = SM.TriDES_Decrypt(encrypt, KEY, IV);
        file_name = "dec_3DES.txt";
    }
    if (alg_flag == "AES")
    {
        decrypt = SM.AES_Decrypt(encrypt, KEY, IV);
        file_name = "dec_AES.txt";
    }

    string dec_res = Encoding.UTF8.GetString(decrypt, 0, decrypt.Length);
    File.WriteAllText(file_name, dec_res);
    textBox4.Text = dec_res;
}

private void Button4_Click(object sender, EventArgs e) // генератор ключей
{
    string file_key = "key.bin";
    string file_IV = "IV.bin";
    textBox5.Clear();
    textBox6.Clear();

    Sym_alg SM = new Sym_alg();

    byte[] k = SM.key_gen();
    byte[] iv = SM.key_gen();

    for (int i = 0; i < k.Length; i++)
        textBox5.Text += k[i].ToString("x") + " ";

    for (int i = 0; i < iv.Length; i++)
        textBox6.Text += iv[i].ToString("x") + " ";

    File.WriteAllBytes(file_key, k);
    File.WriteAllBytes(file_IV, iv);
}
}

```

Asym_alg.cs

```
class Asym_alg
{
    public RSAParameters RSA_KEY_GEN() // генератор ключей для RSA алгоритма
    {
        RSACryptoServiceProvider RSA_key = new RSACryptoServiceProvider();
        return RSA_key.ExportParameters(true);
    }

    public byte[] RSA_Encrypt(byte[] data, RSAParameters RSA_key) // шифратор сообщения
    {
        RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
        byte[] res = new byte[1];

        RSA.ImportParameters(RSA_key); // получаем ключи

        res = RSA.Encrypt(data, true); // шифруем сообщение
        return res;
    }

    public byte[] RSA_Decrypt(byte[] data, RSAParameters RSA_key) // дешифратор сообщения
    {
        RSACryptoServiceProvider RSA = new RSACryptoServiceProvider();
        byte[] res = new byte[1];

        RSA.ImportParameters(RSA_key); // получаем ключ

        res = RSA.Decrypt(data, true); // дешифруем сообщение
        return res;
    }
}
```

Asym.cs

```
public partial class Asym : Form
{
    public Asym()
    {
        InitializeComponent();

        RSAParameters RSA_key = new RSAParameters();
        byte[] b_mess = new byte[1]; // сообщение в байтах
        string[] mess = new string[1]; // сообщение в строках
        byte[] enc = new byte[1]; // шифротекст

        private void button3_Click(object sender, EventArgs e)
        {
            textBox2.Clear();
            File_f F = new File_f();
            string file_name = textBox1.Text;
            if (file_name == "") // проверка на ввод имени файла
                MessageBox.Show("Enter file name.");
            else
            {
                bool fl = File.Exists(file_name); // проверка на существование файла
                if (fl)
                {
                    b_mess = File.ReadAllBytes(file_name);
                    mess = F.Read(file_name);
                    string ms = "";
                    for (int i = 0; i < mess.Length; i++)
                        ms += mess[i];
                    textBox2.Text = ms;
                }
                else
                    MessageBox.Show("File don't exist.");
            }
        }

        private void Button1_Click(object sender, EventArgs e) // шифруем сообщение
        {
            textBox3.Clear();
            Asym_alg AG = new Asym_alg();
            RSA_key = AG.RSA_KEY_GEN();
            enc = AG.RSA_Encrypt(b_mess, RSA_key);
            for (int i = 0; i < enc.Length; i++)
                textBox3.Text += enc[i].ToString("x") + " ";
        }

        private void Button2_Click(object sender, EventArgs e) // дешифруем сообщение
        {
            textBox4.Clear();
        }
    }
}
```

```

        Asym_alg AG = new Asym_alg();
        byte[] dec = AG.RSA_Decrypt(enc, RSA_key);
        string dec_res = Encoding.UTF8.GetString(dec, 0, dec.Length);
        textBox4.Text = dec_res;
    }

```

Dig_key_alg.cs

```

class Dig_key_alg // демонстрационный класс для создания цифровых подписей
{
    public void Key_gen(string Alg_name, string key_file) // функция генерации ключей (ТОЛЬКО ДЛЯ DSA И HMACSHA512)
    {
        if(Alg_name == "DSA")
        {
            DSACng DS = new DSACng(); // создаём объект класса DSACng
            string param = DS.ToXmlString(true); // экспортируем открытый и закрытый ключи в строку XML формата
            using (StreamWriter writer = new StreamWriter(File.Open(key_file, FileMode.Create))) // записываем строку с файл
            {
                writer.Write(param);
            }
        }
        if (Alg_name == "HMACSHA512")
        {
            byte[] key = new byte[64];
            RNGCryptoServiceProvider rng = new RNGCryptoServiceProvider(); // создаём объект специального класса для
            создания ключей
            rng.GetBytes(key); // создаём 64 случайных байта ключа
            File.WriteAllBytes(key_file, key); // записываем ключ в файл
        }
    }

    public byte[] DSA_Sig_Create(byte[] Data, string para_f) // функция для создания DSA подписи
    {
        byte[] Sig = new byte[1]; // подпись
        string param = "";
        using (StreamReader reader = new StreamReader(File.Open(para_f, FileMode.Open))) // считываем параметры из файла
        {
            param = reader.ReadToEnd();
        }

        DSACng DS = new DSACng(); // объект класса DSACng
        DS.FromXmlString(param); // импортируем параметры

        Sig = DS.CreateSignature(Data); // создаём подпись

        return Sig;
    }

    public bool DSA_Sig_Ver(byte[] Data, byte[] Signature, string para_f) // функция для проверки DSA подписи
    {
        string param = "";
        using (StreamReader reader = new StreamReader(File.Open(para_f, FileMode.Open))) // считываем параметры из файла
        {
            param = reader.ReadToEnd();
        }

        DSACng DS = new DSACng();
        DS.FromXmlString(param); // импортируем параметры

        if (DS.VerifySignature(Data, Signature)) // проверяем подпись
        {
            return true;
        }
        else
        {
            return false;
        }
    }

    public byte[] bit_mod(string n_bit, byte[] Sig) // функция модификации подписи (меняет указанный номер бита в подписи)
    {
        int i = Convert.ToInt32(n_bit); // извлекаем номер бита
        BitArray sig_m = new BitArray(Sig); // преобразуем массив байтов в массив битов
        if (sig_m[i] == true) // инвертируем бит
        {
            sig_m[i] = false;
        }
        else
        {
            sig_m[i] = true;
        }

        byte[] ou = new byte[(sig_m.Length - 1) / 8 + 1]; // преобразуем из битов в байты
        sig_m.CopyTo(ou, 0);
        return ou;
    }

    public byte[] ECDSA_Sig_Create(byte[] Data, string para_f) // функция создания ECDSA подписи
    {
        byte[] Sig = new byte[1];
        ECDsaCng ECD = new ECDsaCng(); // создаём объект основного класса
        ECD.HashAlgorithm = CngAlgorithm.Sha256; // указываем алгоритм, на основе которого будем создавать подпись
        File.WriteAllBytes(para_f, ECD.Key.Export(CngKeyBlobFormat.EccPublicBlob)); // экспортируем ключи в файл
        (необходимо, для правильной проверки подписи)
        Sig = ECD.SignData(Data); // создаём подпись

        return Sig;
    }

    public bool ECDSA_Sig_Ver(byte[] Data, byte[] Signature, string para_f) // функция проверки ECDSA подписи

```

```

{
    byte[] key = File.ReadAllBytes(para_f); // получаем ключ из файла
    ECDSAkey ecscKey = new ECDSAkey(CngKey.Import(key, CngKeyBlobFormat.EccPublicBlob)); // создаём объект с заданным
    ключом

    if (ecscKey.VerifyData(Data, Signature)) // проверяем подпись
        return true;
    else
        return false;
}

public byte[] HMAC_Sig_Create(byte[] Data, string para_f) // функция для создания HMAC подписи
{
    byte[] Sig = new byte[1];
    HMACSHA512 hmac = new HMACSHA512(File.ReadAllBytes(para_f)); // считываем параметры из файла и создаём на их основе
    объект класса HMACSHA512 (HMAC на основе SHA512)
    Sig = hmac.ComputeHash(Data); // создаём подпись (подписью является хэш)

    return Sig;
}

public bool HMAC_Sig_Ver(byte[] Data, byte[] Signature, string para_f) // функция проверки HMAC подписи
{
    bool ver = true;
    HMACSHA512 hmac = new HMACSHA512(File.ReadAllBytes(para_f)); // создаём объект класса на основе параметров из файла
    byte[] ver_sig = hmac.ComputeHash(Data); // создаём подпись (хэш)
    for (int i = 0; i < ver_sig.Length; i++) // сравниваем хэши (должны быть равны)
        if (Signature[i] != ver_sig[i])
            ver = false;

    return ver;
}
}

```

Dig_key.cs

```

public partial class Dig_key : Form
{
    public Dig_key()
    {
        InitializeComponent();
        comboBox1.Items.Add("DSA");
        comboBox1.Items.Add("ECDSA");
        comboBox1.Items.Add("HMACSHA512");
        comboBox1.SelectedIndex = 0;
    }

    byte[] b_mess = new byte[1]; // байты сообщения
    string[] mess = new string[1]; // строка с сообщением
    string alg_flag = ""; // название алгоритма
    string para_file_name = "_Paramet.txt"; // левая часть названия файла с параметрами (правой является название алгоритма)
    string out_file_name = "_out.txt"; // левая часть названия файла с подписью (правой является название алгоритма)
    byte[] sig = new byte[1]; // подпись
    private void button1_Click(object sender, EventArgs e)
    {
        File_f F = new File_f();
        textBox2.Clear();
        string file_name = textBox1.Text;
        if (file_name == "") // проверка на ввод имени файла
            MessageBox.Show("Enter file name.");
        else
        {
            bool fl = File.Exists(file_name); // проверка на существование файла
            if (fl)
            {
                b_mess = File.ReadAllBytes(file_name);
                mess = F.Read(file_name);
                string ms = "";
                for (int i = 0; i < mess.Length; i++)
                    ms += mess[i];
                textBox2.Text = ms;
            }
            else
                MessageBox.Show("File don't exist.");
        }
    }

    private void Button3_Click(object sender, EventArgs e) // создание подписи
    {
        textBox3.Clear();
        Dig_key_alg DKA = new Dig_key_alg();

        alg_flag = comboBox1.SelectedItem.ToString();
        if (alg_flag == "DSA")
        {
            sig = DKA.DSA_Sig_Create(b_mess, alg_flag + para_file_name);
        }
    }
}

```

```

        for (int i = 0; i < sig.Length; i++)
            textBox3.Text += sig[i].ToString("x") + " ";
        textBox6.Text = (sig.Length * 8).ToString();
    }
    if (alg_flag == "ECDSA")
    {
        sig = DKA.ECDSA_Sig_Create(b_mess, alg_flag + para_file_name);

        for (int i = 0; i < sig.Length; i++)
            textBox3.Text += sig[i].ToString("x") + " ";
        textBox6.Text = (sig.Length * 8).ToString();
    }
    if (alg_flag == "HMACSHA512")
    {
        sig = DKA.HMAC_Sig_Create(b_mess, alg_flag + para_file_name);

        for (int i = 0; i < sig.Length; i++)
            textBox3.Text += sig[i].ToString("x") + " ";
        textBox6.Text = (sig.Length * 8).ToString();
    }
    File.WriteAllBytes(alg_flag + out_file_name, sig);
}

private void button4_Click(object sender, EventArgs e) // модификация подписи
{
    Dig_key_alg DKA = new Dig_key_alg();
    string b_num = textBox5.Text;
    if (b_num == "" || Convert.ToInt32(b_num) > (sig.Length * 8))
        MessageBox.Show("Не корректный номер бита.");
    else
        sig = DKA.bit_mod(b_num, sig);
}

private void button2_Click(object sender, EventArgs e) // проверка подписи
{
    Dig_key_alg DKA = new Dig_key_alg();
    alg_flag = comboBox1.SelectedItem.ToString();
    if (alg_flag == "DSA")
    {
        if (DKA.DSA_Sig_Ver(b_mess, sig, alg_flag + para_file_name))
            textBox4.Text = "Верный";
        else
            textBox4.Text = "Не верный";
    }
    if (alg_flag == "ECDSA")
    {
        if (DKA.ECDSA_Sig_Ver(b_mess, sig, alg_flag + para_file_name))
            textBox4.Text = "Верный";
        else
            textBox4.Text = "Не верный";
    }
    if (alg_flag == "HMACSHA512")
    {
        if (DKA.HMAC_Sig_Ver(b_mess, sig, alg_flag + para_file_name))
            textBox4.Text = "Верный";
        else
            textBox4.Text = "Не верный";
    }
}

private void Button5_Click(object sender, EventArgs e)
{
    Dig_key_alg DKA = new Dig_key_alg();
    alg_flag = comboBox1.SelectedItem.ToString();
    DKA.Key_gen(alg_flag, alg_flag + para_file_name); // создаём ключ
}
}

```


Hash_alg.cs

```
class Hash_alg
{
    public byte[] SHA_512(byte[] data) // генератор хэша на основе SHA512
    {
        byte[] res = new byte[1];

        SHA512 SHA = new SHA512Managed();

        res = SHA.ComputeHash(data);
        return res;
    }

    public byte[] RIPEMD_160(byte[] data) // генератор хэша на основе RIPEMD160
    {
        byte[] res = new byte[1];

        RIPEMD160 RIPEMD = new RIPEMD160Managed();

        res = RIPEMD.ComputeHash(data);

        return res;
    }
}
```

Hash.cs

```
public partial class Hash : Form
{
    public Hash()
    {
        InitializeComponent();
        comboBox1.Items.Add("SHA512");
        comboBox1.Items.Add("RIPEMD160");
        comboBox1.SelectedIndex = 0;
    }

    byte[] b_mess = new byte[1]; // сообщение в байтах
    string[] mess = new string[1]; // сообщение в строках
    string alg_flag = ""; // название алгоритма
    byte[] hash_arr = new byte[1]; // хэш

    private void Button3_Click(object sender, EventArgs e)
    {
        File_f F = new File_f();
        textBox2.Clear();
        string file_name = textBox1.Text;
        if (file_name == "") // проверка на ввод имени файла
            MessageBox.Show("Enter file name.");
        else
        {
            bool fl = File.Exists(file_name); // проверка на существование файла
            if (fl)
            {
                b_mess = File.ReadAllBytes(file_name);
                mess = F.Read(file_name);
                string ms = "";
                for (int i = 0; i < mess.Length; i++)
                    ms += mess[i];
                textBox2.Text = ms;
            }
            else
                MessageBox.Show("File don't exist.");
        }
    }

    private void Button1_Click(object sender, EventArgs e) // создаём хэш
    {
        textBox3.Clear();
        string file_sha = "SHA512.bin";
        string file_rip = "RIPEMD160.bin";
        Hash_alg hash = new Hash_alg();
        alg_flag = comboBox1.SelectedItem.ToString();
        if (alg_flag == "SHA512")
        {
            hash_arr = hash.SHA_512(b_mess);
            File.WriteAllBytes(file_sha, hash_arr);
        }
        if (alg_flag == "RIPEMD160")
        {
            hash_arr = hash.RIPEMD_160(b_mess);
            File.WriteAllBytes(file_rip, hash_arr);
        }
    }
}
```

```

        for (int i = 0; i < hash_arr.Length; i++)
            textBox3.Text += hash_arr[i].ToString("x") + " ";
    }
}

```

Form1.cs

```

public partial class Form1 : Form
{
    public Form1()
    {
        InitializeComponent();
    }

    // ВЫЗОВ ОКОН
    private void button2_Click(object sender, EventArgs e) // асимметричный алгоритм
    {
        Asym examp = new Asym();
        examp.Show();
    }

    private void Button1_Click(object sender, EventArgs e) // симметричные алгоритмы
    {
        Sym examp = new Sym();
        examp.Show();
    }

    private void Button3_Click(object sender, EventArgs e) // цифровые ключи
    {
        Dig_key examp = new Dig_key();
        examp.Show();
    }

    private void Button4_Click(object sender, EventArgs e) // хэш-функции
    {
        Hash examp = new Hash();
        examp.Show();
    }
}

```

6. Выводы

В ходе выполненной лабораторной было разработано программное средство (демонстрационный стенд), для демонстрации возможностей криптографической библиотеки System.Security.Cryptography.