ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Факультет безопасности информационных технологий

Дисциплина:

«Разработка систем аутентификации и криптографии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

Алгоритмы криптографии и подпись приложений

DES

Выполнили:

Студентка гр. N42514c

Холоденина А.В.

Проверил:

Фёдоров И.Р.

Санкт-Петербург

2020 г.

Оглавление

Цель работы

Часть 1*:* реализация алгоритма шифрования DES.

Требования к реализации:

- необходимо реализовать сам алгоритм (процедуры генерации ключей, шифрования и дешифрования) без использования криптографических библиотек;
- программа должна запускаться в среде Windows, исполняемый файл программы должен иметь расширение .EXE

Часть 2: подпись полученного в первой части файла .EXE

Требования к выполнению:

- необходимо подписать полученный файл .EXE с помощью команд Windows PowerShell (лучше использовать PKI Client)
- при открытии “Свойств” файла .EXE в разделе “Цифровые подписи” мы должны будем увидеть свою подпись.

Описание выбранных средств реализации и обоснование выбора

Для реализации алгоритма шифрования был выбран C#. Это язык программирования, предназначенный для разработки самых разнообразных приложений, предназначенных для выполнения в среде .NET Framework. C# — это объектно- и компонентно-ориентированный язык программирования. C# предоставляет языковые конструкции для непосредственной поддержки такой концепции работы. Благодаря этому C# подходит для создания и применения программных компонентов.

В качестве платформы для реализации была выбрана среда Visual Studio 2019. Она поддерживает выбранный язык программирования и предоставляет возможность использовать Visual C# для удобства настройки интерфейса.

Описание алгоритма

Стандарт шифрования DES (Data Encryption Standard) – был опубликован в 1977 году Национальным бюро стандартов США с целью использования в государственных и правительственных учреждениях для защиты от несанкционированного доступа к важной, но несекретной информации.

Основные достоинства алгоритма DES:

- используется только один ключ длиной 56 битов;
- зашифровав сообщение с помощью одного пакета, для расшифровки вы можете использовать любой другой;
- относительная простота алгоритма обеспечивает высокую скорость обработки информации;
- достаточно высокая стойкость алгоритма.

DES осуществляет шифрование 64-битовых блоков данных с помощью 56-битового ключа. Расшифрование в DES является операцией обратной шифрованию и выполняется путем повторения операций шифрования в обратной последовательности.

Сам процесс шифрования состоит из начальной перестановки, шестнадцати раундах шифрования и конечной перестановки.

1. *Начальная перестановка*. Исходный текст (блок 64 бит)преобразуется с помощью начальной перестановки IP.
2. *Получение 16 ключей по 48 бит из ключа 56 бит.* Ключи *ki* получаются из начального ключа *k* (64 бит = 8 байтов или 8 символов в ASCII) таким образом. Восемь битов, находящих в позициях 8, 16, 24, 32, 40, 48, 56, 64 добавляются в ключ *k* таким образом чтобы каждый байт содержал нечетное число единиц. Это используется для обнаружения ошибок при обмене и хранении ключей. Затем делают перестановку для расширенного ключа (кроме добавляемых битов 8, 16, 24, 32, 40, 48, 56, 64).
3. *Описание функции F*. В функции F находится вся не линейная часть, осуществляется она с помощью S и P преобразований. На каждом этапе биты ключа сдвигаются, и затем из 56 битов ключа выбираются 48 битов с помощью *перестановки со сжатием (PC).* Правая половина входных данных увеличивается до 48 битов с помощью *перестановки с расширением* (*E* или *EP*), а затем объединяется посредством XOR с 48 битами смещенного и переставленного ключа, проходит через восемь 5-блоков, образуя 32 новых бита, и переставляется снова *(P).* Эти четыре операции выполняются функцией раундового преобразования E. Затем результат функции *F* объединяется с левой половиной с помощью XOR.
4. *Описание S-преобразования.* 48-битовый результат сложения расширения правого блока и раундового ключа разбивается на восемь фрагментов но шесть бит, которые подаются на входы соответствующих таблиц замен (S-блоков). У каждого S-блока 6-битовый вход и 4-битовый выход, всего используется восемь различных S-блоков.
5. *Описание P-преобразования*. Поскольку DES построен на схеме Фейстеля, результат перестановки с помощью P-блока объединяется посредством XOR с левой половиной первоначального 64-битового блока. Затем левая и правая половины меняются местами,

и начинается следующий раунд. После последнего раунда DES левая и правая половины местами не меняются.

6. *Конечная перестановка*. Конечная перестановка IP^{-1} действует на T16 и используется для восстановления позиции. Она является обратной к перестановке IP.

При расшифровании данных все действия выполняются в обратном порядке. В 16 циклах расшифрования, в отличие от шифрования с помощью прямого преобразования сетью Фейстеля, здесь используется обратное преобразование сетью Фейстеля.

$$Ri-1=Li$$
$$Li-1=Ri \oplus f(Li,ki)$$

Ключ ki, i=1,…,16, функция f, перестановка IP и IP^{-1} такие же, как и в процессе шифрования.

На данный момент данный алгоритм шифрования признан ненадежным и рекомендовано использовать другие.

Исходный код

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

namespace DES
{
    public partial class Form1 : Form
    {
        private const int sizeOfBlock = 128; //увеличиваем размер блока под unicode
        private const int sizeOfChar = 16; //размер одного символа

        private const int shiftKey = 2; //сдвиг ключа

        private const int quantityOfRounds = 16; //количество раундов

        string[] Blocks; //сами блоки в двоичном формате

        public Form1()
        {
            InitializeComponent();
        }

        //зашифровать
        private void encrypt_Click(object sender, EventArgs e)
        {
            if (textBox1.Text.Length > 0)
            {
                string s = "";

                string key = textBox1.Text;

                StreamReader sr = new StreamReader("in.txt");

                while (!sr.EndOfStream)
                {
                    s += sr.ReadLine();
                }

                sr.Close();

                s = StringToRightLength(s);

                CutStringIntoBlocks(s);

                key = CorrectKeyWord(key, s.Length / (2 * Blocks.Length));
                textBox1.Text = key;
                key = StringToBinaryFormat(key);

                for (int j = 0; j < quantityOfRounds; j++)
                {
                    for (int i = 0; i < Blocks.Length; i++)
```

```
                Blocks[i] = EncodeDES_One_Round(Blocks[i], key);

            key = NextRound(key);
        }

        key = PrevRound(key);

        textBox1.Text = StringFromBinaryToNormalFormat(key);

        string result = "";

        for (int i = 0; i < Blocks.Length; i++)
                result += Blocks[i];

        StreamWriter sw = new StreamWriter("out1.txt");
        sw.WriteLine(StringFromBinaryToNormalFormat(result));
        sw.Close();

        Process.Start("out1.txt");
    }
    else
        MessageBox.Show("Введите ключевое слово!");
}

//расшифровать
private void decrypt_Click(object sender, EventArgs e)
{
    if (textBox1.Text.Length > 0)
    {
        string s = "";

        string key = StringToBinaryFormat(textBoxDecodeKeyWord.Text);

        StreamReader sr = new StreamReader("out1.txt");

        while (!sr.EndOfStream)
        {
            s += sr.ReadLine();
        }

        sr.Close();

        s = StringToBinaryFormat(s);

        CutBinaryStringIntoBlocks(s);

        for (int j = 0; j < quantityOfRounds; j++)
        {
            for (int i = 0; i < Blocks.Length; i++)
                Blocks[i] = DecodeDES_One_Round(Blocks[i], key);

            key = PrevRound(key);
        }

        key = NextRound(key);

        textBox1.Text = StringFromBinaryToNormalFormat(key);

        string result = "";

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

```
            result += Blocks[i];

        StreamWriter sw = new StreamWriter("out2.txt");
        sw.WriteLine(StringFromBinaryToNormalFormat(result));
        sw.Close();

        Process.Start("out2.txt");
    }
    else
        MessageBox.Show("Введите ключевое слово!");
}

//доводим строку до размера, чтобы делилась на sizeOfBlock
private string StringToRightLength(string input)
{
    while (((input.Length * sizeOfChar) % sizeOfBlock) != 0)
        input += "#";

    return input;
}

//разбиение обычной строки на блоки
private void CutStringIntoBlocks(string input)
{
    Blocks = new string[(input.Length * sizeOfChar) / sizeOfBlock];

    int lengthOfBlock = input.Length / Blocks.Length;

    for (int i = 0; i < Blocks.Length; i++)
    {
        Blocks[i] = input.Substring(i * lengthOfBlock, lengthOfBlock);
        Blocks[i] = StringToBinaryFormat(Blocks[i]);
    }
}

//разбиение двоичной строки на блоки
private void CutBinaryStringIntoBlocks(string input)
{
    Blocks = new string[input.Length / sizeOfBlock];

    int lengthOfBlock = input.Length / Blocks.Length;

    for (int i = 0; i < Blocks.Length; i++)
        Blocks[i] = input.Substring(i * lengthOfBlock, lengthOfBlock);
}

//перевод строки в двоичный формат
private string StringToBinaryFormat(string input)
{
    string output = "";

    for (int i = 0; i < input.Length; i++)
    {
        string char_binary = Convert.ToString(input[i], 2);

        while (char_binary.Length < sizeOfChar)
            char_binary = "0" + char_binary;

        output += char_binary;
    }
```

```
    return output;
}

//доводим длину ключа до нужной
private string CorrectKeyWord(string input, int lengthKey)
{
    if (input.Length > lengthKey)
        input = input.Substring(0, lengthKey);
    else
        while (input.Length < lengthKey)
            input = "0" + input;

    return input;
}

//шифрование DES один раунд
private string EncodeDES_One_Round(string input, string key)
{
    string L = input.Substring(0, input.Length / 2);
    string R = input.Substring(input.Length / 2, input.Length / 2);

    return (R + XOR(L, f(R, key)));
}

//расшифровка DES один раунд
private string DecodeDES_One_Round(string input, string key)
{
    string L = input.Substring(0, input.Length / 2);
    string R = input.Substring(input.Length / 2, input.Length / 2);

    return (XOR(f(L, key), R) + L);
}

//XOR двух строк с двоичными данными
private string XOR(string s1, string s2)
{
    string result = "";

    for (int i = 0; i < s1.Length; i++)
    {
        bool a = Convert.ToBoolean(Convert.ToInt32(s1[i].ToString()));
        bool b = Convert.ToBoolean(Convert.ToInt32(s2[i].ToString()));

        if (a ^ b)
            result += "1";
        else
            result += "0";
    }
    return result;
}

//шифрующая функция f. в данном случае это XOR
private string f(string s1, string s2)
{
    return XOR(s1, s2);
}

//вычисление ключа для следующего раунда шифрования. циклический сдвиг >> 2
private string NextRound(string key)
{
    for (int i = 0; i < shiftKey; i++)
```

```
            {
                key = key[key.Length - 1] + key;
                key = key.Remove(key.Length - 1);
            }

            return key;
        }

        //вычисление ключа для следующего раунда расшифровки. циклический сдвиг << 2
        private string PrevRound(string key)
        {
            for (int i = 0; i < shiftKey; i++)
            {
                key = key + key[0];
                key = key.Remove(0, 1);
            }

            return key;
        }

        //переводим строку с двоичными данными в символьный формат
        private string StringFromBinaryToNormalFormat(string input)
        {
            string output = "";

            while (input.Length > 0)
            {
                string char_binary = input.Substring(0, sizeOfChar);
                input = input.Remove(0, sizeOfChar);

                int a = 0;
                int degree = char_binary.Length - 1;

                foreach (char c in char_binary)
                    a += Convert.ToInt32(c.ToString()) * (int)Math.Pow(2, degree--);

                output += ((char)a).ToString();
            }

            return output;
        }


    }
}
```

Демонстрация работы программы

Для открытия программы необходимо запустить файл des.exe.

При запуске программы требуется ввести фразу, которую необходимо зашифровать (рис.1).

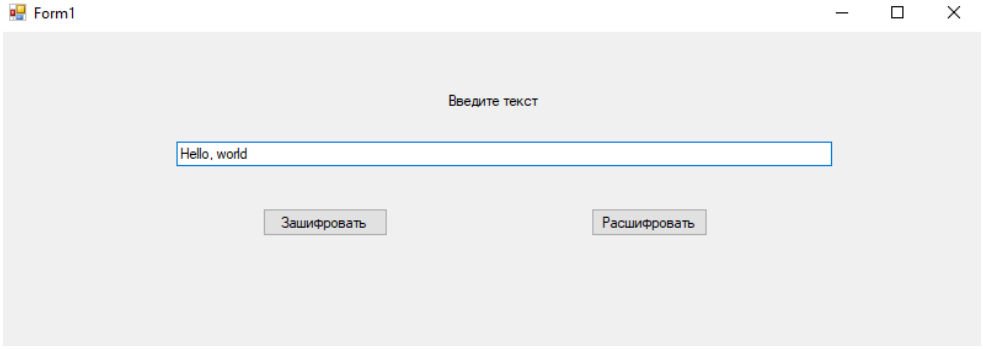

Рис. 1. Запуск программы.

Далее следует нажать кнопку «зашифровать». Произойдет замещение введенного текста зашифрованным (рис.2).

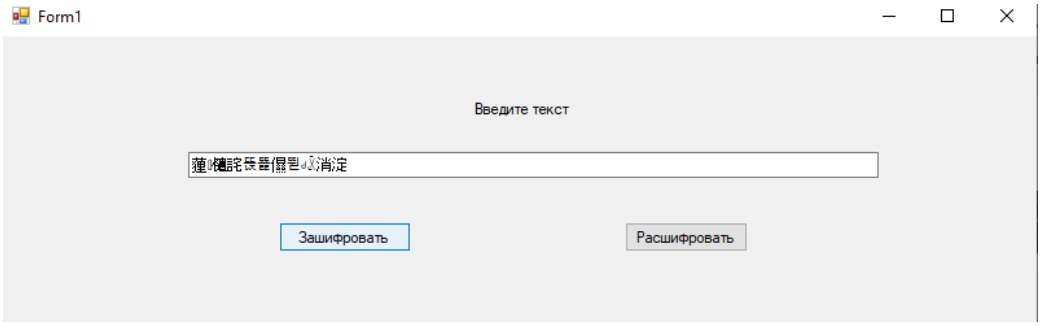

Рис. 2. Шифрование текста.

Для расшифрования текста необходимо нажать кнопку «расшифровать». Программа выведет первоначальный текст (рис.3).

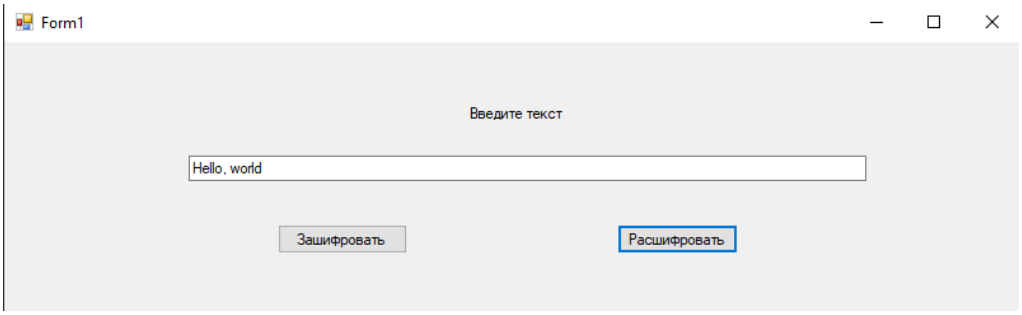

Рис.3. Расшифрование текста.

Подпись exe-файла

Подписать exe файл сертификатом можно осуществить следующим образом:

1. Создать сертификат командой:

```
New-SelfSignedCertificate -Type Custom -Subject "CN=Anna Kholodenina, O=ITMO, C=RU" -KeyUsage DigitalSignature -FriendlyName "Anna Kholodenina" -CertStoreLocation "Cert:\CurrentUser\My"
```

2. Задать переменной cert только что созданный сертификат:

```
$cert=Get-ChildItem -Path cert:\CurrentUser\my -CodeSigningCert
```

3. Подписать exe файл этим сертификатом командой:

```
Set-AuthenticodeSignature des.exe $cert
```

4. Файл подписан (рис.4).

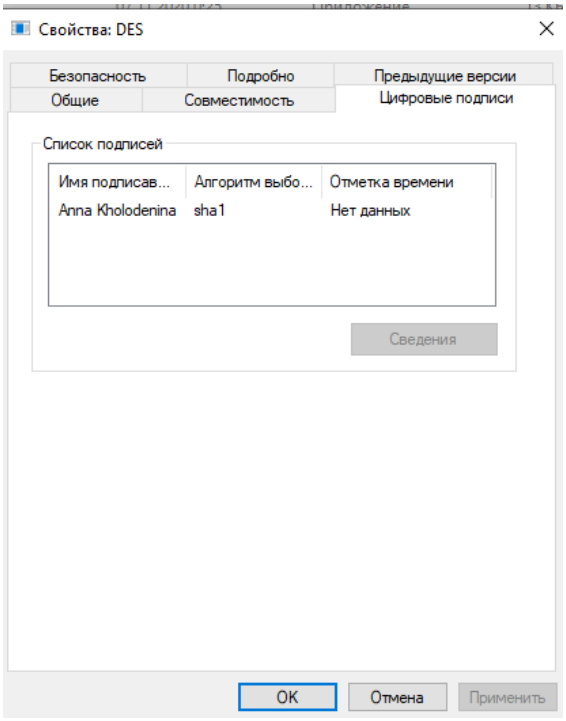

Рис.4. Демонстрация цифровой подписи.

Выводы

В ходе лабораторной работы были изучены основные принципы работы алгоритма DES, а также подпись exe файлов с помощью powershell. В частности, была написана программа на языке C#, реализующая алгоритм DES с графическим интерфейсом, полученная программа была преобразована в .exe файл и подписана с помощью PKI Client.