**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**

**3DES**

**Выполнили:**

Студенты гр. N42514c

Родиманова Е.С.

**Проверил:**

Фёдоров И.Р.

Санкт-Петербург

2020 г.

**Оглавление**

[Цель работы 3](#_Toc55243600)

[Описание выбранных средств реализации и обоснования выбора 3](#_Toc55243601)

[Описание алгоритма 3](#_Toc55243602)

[Исходный код 4](#_Toc55243603)

[Демонстрация работы программы 9](#_Toc55243604)

[Подпись exe-файла 10](#_Toc55243605)

[Выводы 11](#_Toc55243606)

# Цель работы

**-** реализовать алгоритм шифрования **3DES (**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;

- подпись полученного файла .exe (при открытии «Свойств» файла .exe в разделе «Цифровые подписи» можно увидеть свою подпись).

# Описание выбранных средств реализации и обоснования выбора

В качестве языка программирования был выбран C#. Этот язык является объектно-ориентированным языком программирования, разработанный компанией Microsoft в качестве языка для разработки приложений для платформы Microsoft .NET Framework. В свою очередь, .NET Framework - это программная платформа, т. е. некая "среда выполнения", в которой должен работать код, написанный для данной платформы.

В качестве среды разработки будет использована Microsoft Visual Studio 2015. Эта среда удобна для исполнения данного задания, так как в ней можно достаточно просто нанести нужный интерфейс и прописать действия, а также описать сам алгоритм.

# Описание алгоритма

Алгоритм DES разработан командой математиков научной лаборатории IBM, в которую входил Фейстель. Первая версия шифра получила имя «Люцифер», но затем он был модифицирован и в результате принят как официальный алгоритм шифрования данных (DEA). На протяжении более двадцати лет он оставался мировым стандартом, прежде чем его сменил Triple DES.

Рассмотрим, как работает алгоритм шифрования DES. Для этого необходимо вспомнить работу сети Фейстеля. DES — это сеть Фейстеля из 16 раундов с симметричными ключами шифрования. Длина блока текста — 64 бита, длина раундового ключа — 48 бит. Пройдем основные этапы шифрования DES:

1. Текст, как и при любом другом шифровании, разбивается на блоки по 64 бита.
2. Из 56-битного ключа генерируется 16 48-битных раундовых ключей.
3. Каждый блок подвергается перестановке, то есть все биты входного блока перемешиваются согласно определенной таблице.
4. Блок расщепляется на половинки и поступает в сеть Фейстеля, где прокручивается 16 раундов.
5. Соединяются половинки.
6. И еще одна перестановка.

Triple DES – это, по сути, тройной DES. В нем возникла необходимость, так как 56-битный ключ DES был уязвим к брутфорсу и с ростом вычислительных мощностей эта проблема вставала все острее. Используя доступную сегодня технологию, можно проверить один миллион ключей в секунду. Это означает, что потребуется более чем две тысячи лет, чтобы перебором дешифровать DES, используя компьютер только с одним процессором.

Чтобы решить проблему быстрого поиска ключа, умные зарубежные криптографы предложили использовать два ключа и применять DES дважды. Однако двойной DES оказался уязвим к атаке «встреча посередине». Чтобы реализовать эту атаку, злоумышленнику необходимо иметь открытый и соответствующий ему зашифрованный текст. Злоумышленник шифрует открытый текст на всех возможных ключах, записывая результаты в таблицу 1. Затем расшифровывает зашифрованный текст со всеми возможными ключами и записывает результат в таблицу 2. Далее злоумышленник ищет в таблицах 1 и 2 совпадения.

Атака данного типа заключается в переборе ключей на стороне шифрованного и открытого текста и требует примерно в четыре раза больше вычислений, чем перебор обычного ключа DES, и довольно много памяти для хранения промежуточных результатов. Тем не менее на практике атака осуществима, что делает алгоритм Double DES непригодным.

Совсем иначе дела обстоят с Triple DES. Использование трех ключей и применение алгоритмов в указанной последовательности продлило DES жизнь еще на несколько лет.

Существует 3 типа алгоритма 3DES:

* DES-EEE3: Шифруется три раза с тремя разными ключами (операции шифрование-шифрование-шифрование).P = E k 1 − 1 ( E k 2 − 1 ( E k 3 − 1 ( C ) ) ) {\displaystyle P=E\_{k\_{1}}^{-1}(E\_{k\_{2}}^{-1}(E\_{k\_{3}}^{-1}(C)))}
* DES-EDE3: 3DES операции шифровка-расшифровка-шифровка с тремя разными ключами.
* DES-EEE2 и DES-EDE2: Как и предыдущие, за исключением того, что на первом и третьем шаге используется одинаковый ключ.

Самая популярная разновидность 3DES — это DES-EDE3. В данной реализации представлен алгоритм DES-EEE3.

# Исходный код

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;

namespace \_3DES

{

public partial class Form1 : Form

{

private const int sizeBlock = 128; //в DES размер блока 64 бит, но поскольку в unicode символ в два раза длинее, то увеличим блок тоже в два раза

private const int sizeChar = 16; //размер одного символа

private const int shiftKey = 2; //сдвиг ключа

private const int quantityRounds = 16; //количество раундов

String[] blocks; //блоки в двоичном формате

String str = ""; //введенное сообщение

String key1 = ""; //певый ключ

String key2 = ""; //второй ключ

String key3 = ""; //третий ключ

String key1Temp = ""; //для хранения первого ключа

String key2Temp = ""; //для хранения второго ключа

String key3Temp = ""; //для хранения третьего ключа

String keyTemp = "";

public Form1()

{

InitializeComponent();

}

/\*Шифрование сообщения\*/

private void button1\_Click(object sender, EventArgs e)

{

String result = "";

readText();

/\*Шифрование первым ключом\*/

key1 = keyGen();

result = encode(key1, str);

key1Temp = keyTemp;

/\*Дешифрование вторым ключом\*/

key2 = keyGen();

result = encode(key2, result);

key2Temp = keyTemp;

/\*Шифрование третьим ключом\*/

key3 = keyGen();

result = encode(key3, result);

key3Temp = keyTemp;

/\*Вывод результата\*/

textBox2.Text = result;

}

/\*Шифрование с помощью ключа исходной информации\*/

private string encode(String key, String str)

{

String result = "";

cutStringBlocks(str);

key = correctKeyWord(key, str.Length / (2 \* blocks.Length));

key = stringBinaryFormat(key);

for (int j = 0; j < quantityRounds; j++)

{

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

blocks[i] = encodeDESRound(blocks[i], key);

key = keyNextRound(key);

}

key = keyPrevRound(key);

keyTemp = stringBinaryNormalFormat(key);

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

{

result += blocks[i];

}

return stringBinaryNormalFormat(result);

}

/\*Дешифрование при помощи ключа зашифрованного сообщения\*/

private string decode(String key, String keyTemp, String str)

{

String result = "";

key = stringBinaryFormat(keyTemp);

str = stringBinaryFormat(str);

cutBinaryStringBlocks(str);

for (int j = 0; j < quantityRounds; j++)

{

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

blocks[i] = decodeDESRound(blocks[i], key);

key = keyPrevRound(key);

}

key = keyNextRound(key);

keyTemp = stringBinaryNormalFormat(key);

result = "";

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

result += blocks[i];

return stringBinaryNormalFormat(result);

}

/\*Дешифрование сообщения\*/

private void button2\_Click(object sender, EventArgs e)

{

readText();

String result = "";

/\*Дешифрование третьим ключом\*/

result = decode(key3, key3Temp, str);

key3Temp = keyTemp;

/\*Шифрование вторым ключом\*/

result = decode(key2, key2Temp, result);

key2Temp = keyTemp;

/\*Дешифрование первым ключом\*/

result = decode(key1, key1Temp, result);

key1Temp = keyTemp;

/\*Вывод результата\*/

textBox2.Text = result;

}

/\*Считывание сообщения из текстбокса\*/

private void readText()

{

str = textBox1.Text.ToString();

if (str == "")

{

MessageBox.Show("Введите сообщение, которое необходимо зашифровать!");

}

}

/\*Генерация ключа\*/

private string keyGen()

{

Random random = new Random();

return random.Next(10000, 100000).ToString();

}

/\* Метод, доводящий строку до такого размера, чтобы она делилась на sizeBlock.

\* Размер увеличивается с помощью добавления к исходной строке символа "#"

\*/

private string stringRightLength(string input)

{

while (((input.Length \* sizeChar) % sizeBlock) != 0)

input += "#";

return input;

}

/\*Метод, разбивающий строку в обычном (символьном) формате на блоки\*/

private void cutStringBlocks(string input)

{

blocks = new string[(input.Length \* sizeChar) / sizeBlock];

int lengthBlock = input.Length / blocks.Length;

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

{

blocks[i] = input.Substring(i \* lengthBlock, lengthBlock);

blocks[i] = stringBinaryFormat(blocks[i]);

}

}

/\*Метод, разбивающий строку в двоичном формате на блоки\*/

private void cutBinaryStringBlocks(string input)

{

blocks = new string[input.Length / sizeBlock];

int lengthBlock = input.Length / blocks.Length;

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

blocks[i] = input.Substring(i \* lengthBlock, lengthBlock);

}

/\*Метод, переводящий строку в двоичный формат\*/

private string stringBinaryFormat(string input)

{

string output = "";

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

{

string charBinary = Convert.ToString(input[i], 2);

while (charBinary.Length < sizeChar)

charBinary = "0" + charBinary;

output += charBinary;

}

return output;

}

/\*Метод, доводящий длину ключа до нужной длины\*/

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;

}

/\*Один раунд шифрования алгоритмом DES\*/

private string encodeDESRound(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)));

}

/\*Один раунд расшифровки алгоритмом DES\*/

private string decodeDESRound(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\*/

private string f(string s1, string s2)

{

return XOR(s1, s2);

}

/\*Вычисление ключа для следующего раунда шифрования DES\*/

private string keyNextRound(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 DES\*/

private string keyPrevRound(string key)

{

for (int i = 0; i < shiftKey; i++)

{

key = key + key[0];

key = key.Remove(0, 1);

}

return key;

}

/\*Метод, переводящий строку с двоичными данными в символьный формат\*/

private string stringBinaryNormalFormat(string input)

{

string output = "";

while (input.Length > 0)

{

string char\_binary = input.Substring(0, sizeChar);

input = input.Remove(0, sizeChar);

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 начала необходимо ввести исходное сообщение, чтобы можно было его зашифровать. Для этого необходимо написать нужный текст в поле «Ввод текста» и нажать кнопку «Зашифровать». Результат работы алгоритма отобразится в поле «Вывод текста» (Рисунок 1). Если пользователь не ввел никаких исходных данных, то отобразится ошибка (Рисунок 3).

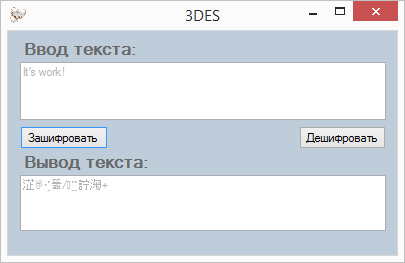


Рисунок 1 – Демонстрация работы кнопки «Зашифровать»

Для проверки правильности работы данного алгоритма можно скопировать данные, которые получились в результате работы из поля «Вывод текста» и вставить его в поле «Ввод текста». После чего необходимо нажать кнопку «Дешифровать». На Рисунке 2 можно увидеть, что в ходе таких манипуляций текст получился одинаковый.

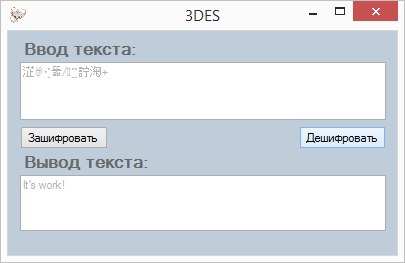


Рисунок 2 – Демонстрация работы кнопки «Дешифровать»

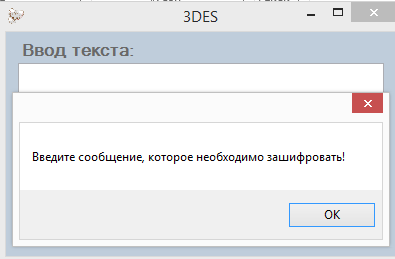


Рисунок 3 – Вывод ошибки в случае отсутствия нужной информации для работы программы

# Подпись exe-файла

Еще одной часть из лабораторной работы было подписание полученного файла .exe. На Рисунке 4 видно, что при открытии «Свойств» файла .exe в разделе «Цифровые подписи» стоит подпись. Эта часть задания была выполнена в PowerShall. Для этого потребовалось создать самоподписанный сертификат, затем задать переменной cert только что созданный сертификат и подписать exe-файл этим сертификатом при помощи команды.

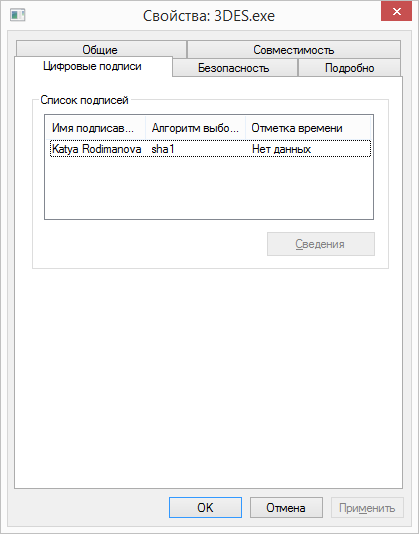


Рисунок 4 – Подпись exe-файла

# Выводы

В ходе лабораторной работы были изучены основные принципы работы алгоритма 3DES, а также подпись exe-файлов с помощью powershell. В частности, была написана программа на языке C#, реализующая заданный алгоритм с графическим интерфейсом, скрипт был преобразован в .exe файл и подписан при помощи команд, выполненных в PowerShall.