

**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ТАРАСА ШЕВЧЕНКА**



ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра прикладних інформаційних систем

Звіт до лабораторної роботи №10

3 курсу

«Безпека мереж і комп'ютерних систем»

*студента 2 курсу
групи ПП-22
спеціальності 122 «Комп'ютерні науки»
ОП «Прикладне програмування»
Шевлюк Вікторії Віталіївни*

*Перевірів:
д.т.н, професор
Сайко В. Г.*

Київ 2022

Тема: Блокові симетричні алгоритми шифрування

Мета: Ознайомитися з основними поняттями, присвяченими принципам функціонування блокових алгоритмів. Дослідити структуру шифру Фейстеля.

Завдання:

1. Вивчити основні теоретичні положення стосовно блокових криптоалгоритмів.

2. Реалізувати програмно блоковий шифр, заснований на шифрі Фейстеля:

- відкритий текст отримується з текстового файлу "open.txt".

Розмір блоку вхідних даних для шифрування дорівнює $2w$ біт;

- ключ задається в тілі програми. Розмір ключа складає N біт;
- число раундів обробки дорівнює n ;
- шифротекст виводиться на екран.

Параметри w, n, N , а також алгоритм обчислення підключів і функція раунду вибираються згідно з варіантом, виданим викладачем.

Хід роботи:

Напишемо мовою C# програму, що буде реалізовувати шифр Фейстеля, що буде працювати наступним чином:

відкритий текст отримується з текстового файлу "open.txt". Розмір блоку вхідних даних для шифрування дорівнює $2w$ біт; ключ задається в тілі програми. Розмір ключа складає N біт; число раундів обробки дорівнює n ; шифротекст виводиться на екран.

Нижче наведений код моєї програми:

```

using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace lab10
{
    Ссылка: 0
    class Program
    {
        Ссылка: 2
        private static byte Encrypt(byte msg, Func<byte, byte, byte> FunctionF, byte[] keys)
        {
            byte step = msg;
            for (int i = 0; i < keys.Length; i++)
            {
                step = FeistelStep(step, keys[i], FunctionF);
            }

            return step;
        }

        Ссылка: 2
        private static byte Decrypt(byte msg, Func<byte, byte, byte> FunctionF, byte[] keys)
        {
            byte step = msg;
            step = InversionLR(step);
            for (int i = keys.Length - 1; i >= 0; i--)
            {
                step = FeistelStep(step, keys[i], FunctionF);
            }
            step = InversionLR(step);

            return step;
        }
    }
}

```

Ссылка: 4

```
private static byte FunctionF(byte x, byte key)
{
    return Xor(x, key);
}
```

Ссылка: 2

```
private static byte FeistelStep(byte msg, byte key, Func<byte, byte, byte> FunctionF)
{
    var R = GetR(msg);
    var L = GetL(msg);

    var funcResult = OperateR(R, key, FunctionF);

    var xorResult = OperateL(L, funcResult, Xor);

    var finalResult = InversionLR(xorResult, R);

    return finalResult;
}
```

Ссылка: 3

```
private static byte Xor(byte x, byte y)
{
    return (byte)((int)x ^ (int)y);
}
```

Ссылка: 2

```
private static byte GetR(byte x)
{
    var temp = (byte)(((int)x) << 4);
    return (byte)(((int)temp) >> 4);
}
```

Ссылка: 2

```
private static byte GetL(byte x)
{
    var temp = (byte)(((int)x) >> 4);
    return (byte)(((int)temp) << 4);
}
```

Ссылка: 0

```
private static byte EnsureKeyHas4Bits(byte key)
{
    return (byte)((int)key % 16);
}
```

Ссылка: 1

```
public static void EncryptFile(string fileInPath, string fileOutPath, Func<byte, byte, byte> FunctionF, byte[] keys)
{
    byte[] file = File.ReadAllBytes(fileInPath);
    byte[] encFile = new byte[file.Length];
    for (int i = 0; i < file.Length; i++)
    {
        encFile[i] = Encrypt(file[i], FunctionF, keys);
    }
    File.WriteAllBytes(fileOutPath, encFile);
}
```

Ссылка: 1

```
public static void DecryptFile(string fileInPath, string fileOutPath, Func<byte, byte, byte> FunctionF, byte[] keys)
{
    byte[] file = File.ReadAllBytes(fileInPath);
    byte[] decFile = new byte[file.Length];
    for (int i = 0; i < file.Length; i++)
    {
        decFile[i] = Decrypt(file[i], FunctionF, keys);
    }
    File.WriteAllBytes(fileOutPath, decFile);
}
```

Ссылка: 0

```
static void Main(string[] args)
{
    byte msg = 150;
    PrintOutByte("msg: ", msg);

    byte[] keys = { 15, 1, 19, 3, 9 };

    var encrypted = Encrypt(msg, FunctionF, keys);
    PrintOutByte("enc: ", encrypted);

    var decrypted = Decrypt(encrypted, FunctionF, keys);
    PrintOutByte("dec: ", decrypted);

    EncryptFile("plaintext.txt", "encoded.txt", FunctionF, keys);
}
```

ссылка: 0

```
private static byte EnsureKeyHas4Bits(byte key)
{
    return (byte)((int)key % 16);
}
```

ссылка: 1

```
public static void EncryptFile(string fileInPath, string fileOutPath, Func<byte, byte, byte> FunctionF, byte[] keys)
{
    byte[] file = File.ReadAllBytes(fileInPath);
    byte[] encFile = new byte[file.Length];
    for (int i = 0; i < file.Length; i++)
    {
        encFile[i] = Encrypt(file[i], FunctionF, keys);
    }
    File.WriteAllBytes(fileOutPath, encFile);
}
```

ссылка: 1

```
public static void DecryptFile(string fileInPath, string fileOutPath, Func<byte, byte, byte> FunctionF, byte[] keys)
{
    byte[] file = File.ReadAllBytes(fileInPath);
    byte[] decFile = new byte[file.Length];
    for (int i = 0; i < file.Length; i++)
    {
        decFile[i] = Decrypt(file[i], FunctionF, keys);
    }
    File.WriteAllBytes(fileOutPath, decFile);
}
```

ссылка: 0

```
static void Main(string[] args)
{
    byte msg = 150;
    PrintOutByte("msg: ", msg);

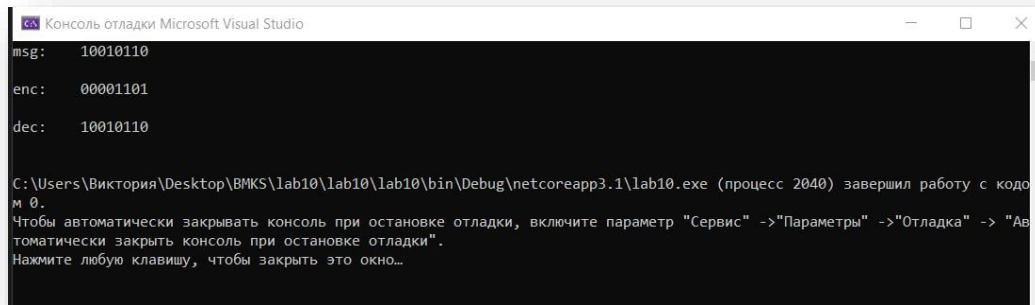
    byte[] keys = { 15, 1, 19, 3, 9 };

    var encrypted = Encrypt(msg, FunctionF, keys);
    PrintOutByte("enc: ", encrypted);

    var decrypted = Decrypt(encrypted, FunctionF, keys);
    PrintOutByte("dec: ", decrypted);

    EncryptFile("plaintext.txt", "encoded.txt", FunctionF, keys);
}
```

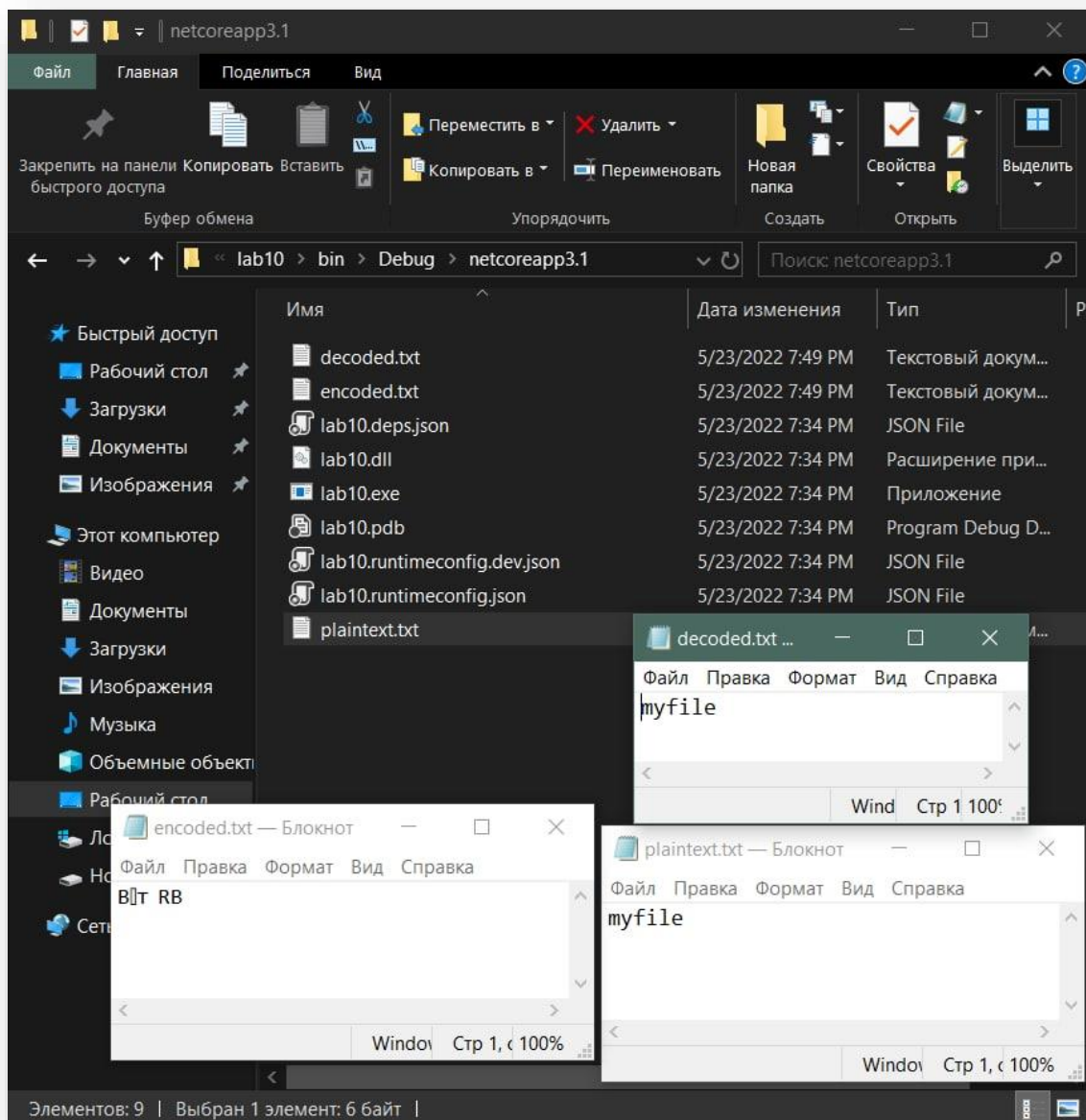
Результат работы програми:



Консоль отладки Microsoft Visual Studio

```
msg: 10010110
enc: 00001101
dec: 10010110
```

C:\Users\Виктория\Desktop\BMKS\lab10\lab10\bin\Debug\netcoreapp3.1\lab10.exe (процесс 2040) завершил работу с кодом 0.
Чтобы автоматически закрывать консоль при остановке отладки, включите параметр "Сервис" -> "Параметры" -> "Отладка" -> "Автоматически закрывать консоль при остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...



Висновок: під час цієї лабораторної роботи я ознайомила з основними поняттями, присвяченими принципам функціонування блокових алгоритмів і дослідила структуру шифру Фейстеля. **Мережа Фейстеля** (конструкція Фейстеля) — різновид блочного шифру з певною ітеративною структурою. Багато сучасних алгоритмів використовують мережу Фейстеля як основу.

Контрольні питання:

1. Які принципи блокового шифрування ви знаєте?

Обробка блоку декількох байт за одну ітерацію (як правило 8 або 16)

2. У чому відмінність між блоковими і потоковими шифрами?

Потоковий шифрує потік байтів, блочний шифрує блоки з декількох байтів за ітерацію

3. Які вимоги висуваються до функції шифрування для блокових шифрів?

До функції блокового шифрування E висуваються наступні вимоги:

Функція E має бути зворотною.

Не повинно існувати інших методів отримання повідомлення (X) по відомому блоку (Z) , окрім як повним перебором ключів K .

Не повинно існувати інших методів визначення, яким ключем було зашифроване повідомлення, окрім як повним перебором ключів

4. На чому заснована криптостійкість блокових шифрів?

Щоб надати необхідну стійкість шифру, шифрувальна функція працює циклічно: це називається циклічним кодуванням

5. Які алгоритми блокового шифрування вам відомі?

Ітераційні блочні шифри, SP-сітки, шифр Фейстеля

6. Опишіть структуру шифру Фейстеля.

На вхід алгоритму шифрування подається блок відкритого тексту завдовжки $2w$ бітів і ключ K . Блок відкритого тексту розділяється на дві рівні частини, L_0 й R_0 , які послідовно проходять через n раундів обробки, а потім об'єднуються знову для отримання блоку шифрованого тексту відповідної довжини. Для раунду i як вхідні дані виступають L_{i-1} и R_{i-1} , отримані на виході попереднього раунду, і підключ K_i , що обчислюється по загальному ключу K . Усі підключі відрізняються як від загального ключа, так і один від одного. Усі раунди обробки проходять за однією і тією ж схемою. Спочатку для лівої половини блоку даних виконується операція підстановки. Вона полягає в застосуванні до правої половини блоку даних деякої функції раунду F і в наступному додаванні отриманого результату з лівою половиною блоку даних за допомогою операції XOR. Для усіх раундів функція F має одну і ту ж структуру, але залежить від параметра – підключа раунда K_i . Після підстановки виконується перестановка, дві половини блоку даних міняються місцями. Уся ця структура в цілому є частинним випадком так званої підстановлювально-перестановочної схеми, запропонованої Шеноном. Після закінчення останнього раунду виконується ще одна перестановка, яка, по суті, відміняє перестановку, виконану в останньому раунді. Тому, знехтувавши лише зовнішньою узгодженістю представлення, ці дві перестановки зі схеми можна видалити.

7. Дайте визначення поняттям дифузії і конфузії.

Суть дифузії полягає в розсіянні статистичних особливостей відкритого тексту по широкому діапазону статистичних характеристик шифрованого тексту. Це досягається тим, що значення кожного елементу відкритого тексту впливає на значення багатьох елементів шифрованого тексту, або будь-який з елементів шифрованого тексту залежить від безлічі елементів відкритого тексту.

Що стосується конфузії, то перед нею ставиться завдання в максимальній мірі ускладнити статистичний взаємозв'язок між шифрованим

текстом і ключем з метою протистояння спробам визначити ключ. Це досягається використанням складних підстановочних алгоритмів: прості лінійні подстановочні функції збільшують складність алгоритму лише в незначній мірі

8. Які параметри шифру Фейстеля вам відомі?

Ключ, число раундів обробки, розмір вхідних даних та ключа