**sМИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

по дисциплине «Основы защиты информации»

**Блочные алгоритмы шифрования**

Выполнил: студент гр. ИТП-31

Расшивалов Н.И.

Принял: Доцент

Кравченко О.А.

Гомель 2021

**Цель работы:** освоить навыки применения классических блочных методов шифрования данных.

**Задание**

Разработать приложение, выполняющее шифрование и дешифрование текста в соответствии с заданным алгоритмом (табл.1).

Ключ выбрать самостоятельно.

Протестировать работу приложения на не менее чем трех текстах от 10 до 20 символов.

Таблица 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Вариант | Вид сети | Образующая функция | Размер входного блока, бит | Число раундов | Расчет параметров сети |
| 2 | Абсолютно симметричная сеть Файстеля | X’=P-box(X) | 64 | 32 | Vi для четных раундов определяет использование первой строки таблицы, для нечетных - второй |

**Ход работы**

В ходе выполнения был создан проект *WPF* для реализации через интерфейс шифрования и дешифрования текста. Для тестирования разработанных классов был создан проект модульного теста.

Класс *Feistel* реализует шифрование и дешифрование текста с помощью блочного алгоритма Файстеля. Результат работы разработанного класса представлен на рисунках 1 - 2. Результаты модульного тестирования представлены на рисунке 3.

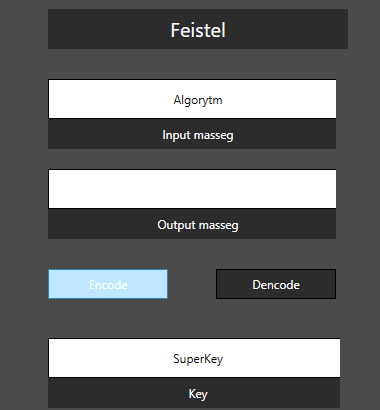


Рисунок1 –Результат работы класса *Feistel*.

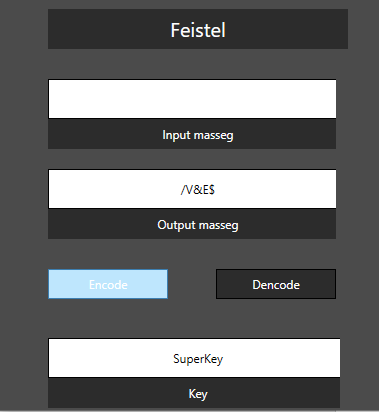


Рисунок2 –Результат работы класса *Feistel*.

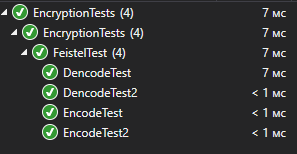


Рисунок 3 –Результаты модульного тестирования.

**Вывод:** в процессе выполнения данной лабораторной работы были изучены основные навыки применения блочного алгоритма. Полученные знания были применены для разработки приложения шифрования и дешифрования.

**Приложение:**

**Листинг Feistel.cs:**

using System;

using System.Collections.Generic;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Encryption

{

public static class Feistel

{

public static string Encode(string message, string key)

{

return Code(message, key);

}

public static string Dencode(string message, string key)

{

return Code(message, key);

}

private static string Code(string message, string key)

{

byte[] messageBytes = Encoding.Default.GetBytes(message);

byte[] keyBytes = Encoding.Default.GetBytes(key);

int diff = messageBytes.Length % 8;

if (diff != 0)

{

byte[] temp = new byte[messageBytes.Length + (8 - diff)];

Array.Copy(messageBytes, temp, messageBytes.Length);

messageBytes = temp;

}

byte[] result= new byte[messageBytes.Length];

for (int i = 0; i < messageBytes.Length; i = i + 8)

{

byte[] block = new byte[8];

Array.Copy(messageBytes, i, block, 0, 8);

for (int j = 0; j < 16; j++)

{

byte[] leftBlock = new byte[4];

Array.Copy(block, leftBlock, 4);

byte[] rightBlock = new byte[4];

Array.Copy(block, 4, rightBlock, 0, 4);

byte[] keyPart = new byte[4];

Array.Copy(keyBytes, keyPart, 4);

keyPart = keyGen(keyBytes, j);

Pbox(block);

block = CodeBlock(leftBlock, rightBlock, keyPart, true);

}

for (int j = 15; j >= 0; j--)

{

byte[] leftBlock = new byte[4];

Array.Copy(block, leftBlock, 4);

byte[] rightBlock = new byte[4];

Array.Copy(block, 4, rightBlock, 0, 4);

byte[] keyPart = new byte[4];

Array.Copy(keyBytes, keyPart, 4);

keyPart = keyGen(keyBytes, j);

if (j != 0)

{

Pbox(block);

block = CodeBlock(leftBlock, rightBlock, keyPart, false);

}

else

block = CodeBlock(leftBlock, rightBlock, keyPart, true);

}

Array.Copy(block, 0, result, i, block.Length);

}

return Encoding.Default.GetString(result);

}

private static byte[] Pbox(byte[] block)

{

byte tmp = block[block.Length - 1];

for (int i = block.Length - 1; i > 0; i--)

{

block[i] = block[i - 1];

}

block[0] = tmp;

return block;

}

private static byte[] CodeBlock(byte[] leftBlock, byte[] rightBlock, byte[] key, bool isLast)

{

int intLeftBlock = BitConverter.ToInt32(leftBlock, 0);

int intKey = BitConverter.ToInt32(key, 0);

intLeftBlock = intLeftBlock ^ intKey;

leftBlock = BitConverter.GetBytes(intLeftBlock);

byte[] result = new byte[8];

if (!isLast)

{

Array.Copy(rightBlock, result, 4);

Array.Copy(leftBlock, 0, result, 4, 4);

}

else

{

Array.Copy(leftBlock, result, 4);

Array.Copy(rightBlock, 0, result, 4, 4);

}

return result;

}

private static byte[] keyGen(byte[] key, int i)

{

byte[] tmp = new byte[4];

Array.Copy(key, tmp, 4);

int left = BitConverter.ToInt32(tmp, 0);

Array.Copy(key, 4, tmp, 0, 4);

int right = BitConverter.ToInt32(tmp, 0);

int l\_l = left << (i \* 3);

int r\_r = right >> (32 - i \* 3);

left = l\_l + r\_r;

return BitConverter.GetBytes(left);

}

}

}

**Листинг FeistelTest.cs:**

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;

using Encryption;

namespace EncryptionTests

{

[TestClass]

public class FeistelTest

{

[TestMethod]

public void EncodeTest()

{

string message = "qwertyqwertyoppo";

string key = "qwertyui";

string expected = "‑ъ'qо\"vы?tя)";

Assert.AreEqual(expected, Feistel.Encode(message, key));

}

[TestMethod]

public void DencodeTest()

{

string message = "‑ъ'qо\"vы?tя)";

string key = "qwertyui";

string expected = "qwertyqwertyoppo";

Assert.AreEqual(expected, Feistel.Dencode(message, key));

}

[TestMethod]

public void EncodeTest2()

{

string message = "ahfdsdgfdstdrfgrser";

string key = "zxcvbnmj";

string expected = "ЯЛ ГНЗ!БЮЙ ЧИЬ3Б¬ЇGҐЯК5Ґ";

Assert.AreEqual(expected, Feistel.Encode(message, key));

}

[TestMethod]

public void DencodeTest2()

{

string message = "ЯЛ ГНЗ!БЮЙ ЧИЬ3Б¬ЇGҐЯК5Ґ";

string key = "zxcvbnmj";

string expected = "ahfdsdgfdstdrfgrser\0\0\0\0\0";

Assert.AreEqual(expected, Feistel.Dencode(message, key));

}

}

}

**Листинг MainWindow.xaml.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using Encryption;

namespace Lab3

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

private void bEF\_Click(object sender, RoutedEventArgs e)

{

textOut.Text = Feistel.Encode(textIn.Text, textKey.Text);

textIn.Text = "";

}

private void bDF\_Click(object sender, RoutedEventArgs e)

{

textIn.Text = Feistel.Encode(textOut.Text, textKey.Text);

textOut.Text = "";

}

}

}