

УО «Белорусский государственный университет информатики и радиоэлектроники»
Кафедра ПОИТ

Отчет по лабораторной работе №2
по предмету
Теория Информации
Вариант 14

Выполнил:
Бетень К.С.

Проверил:
Болтак С.В.

Группа 351001

Минск 2025

Задание:

Реализовать систему потокового шифрования и дешифрования для файла с любым содержимым с помощью генератора ключевой последовательности на основе линейного сдвигового регистра с обратной связью LFSR1 (размерность регистра **36**). Начальное состояние регистра ввести с клавиатуры. Поле для ввода состояния регистра должно игнорировать любые символы кроме 0 и 1. Вывести на экран сгенерированный ключ (последовательность из 0 и 1), исходный файл и зашифрованный файл в двоичном виде. Программа не должна быть написана в консольном режиме. Результат работы программы – зашифрованный/расшифрованный файл.

Примитивный многочлен

$$x^{36} + x^{11} + 1$$

Работа с файлами

Взятие исходного файла:

Исходный текст
Первые 8 байт: 0111001101110100011001010110000101101101011010010110010000111010
Последние 8 байт: 0011011101001000010101010010110101001101010110100101001101001011
Ввести исходный текст из файл

Загрузка результата файла:

Сгенерированный результат
Первые 8 байт: 1000110010001011100110101001111010011000001111000011000101101001
Последние 8 байт: 0011110101010110011101110010101101000111010001000101000101001101
Сохранить сгенерированный результат в файл

Общий внешний вид страницы:

LFSR	
Состояние регистра (36 бита)	Сгенерированный ключ
<input type="text"/>	<input type="text"/>
Длина введенных состояний: 0	Длина сгенерированного ключа: 0
Исходный текст	Сгенерированный результат
<input type="text"/>	<input type="text"/>
Ввести исходный текст из файл	Сохранить сгенерированный результат в файл
Шифрование	Дешифрование

Тесты

1. Простой ключ, небольшой текст.

Ключ: 1111111111111111111111111111111111

Исходный текст: 0011000100110100

Шифротекст: 1100111011001011

Шифрование:

Состояние регистра (36 бита)	Сгенерированный ключ
11111111111111111111111111111111	11111111111111111111111111111111
Длина введенных состояний: 36	Длина сгенерированного ключа: 36

Исходный текст	Сгенерированный результат
0011000100110100	1100111011001011
Ввести исходный текст из файл	Сохранить сгенерированный результат в файл

Шифрование	Дешифрование
------------	--------------

Дешифрирование:

Состояние регистра (36 бита)	Сгенерированный ключ
11111111111111111111111111111111	11111111111111111111111111111111
Длина введенных состояний: 36	Длина сгенерированного ключа: 36

Исходный текст	Сгенерированный результат
1100111011001011	0011000100110100
Ввести исходный текст из файл	Сохранить сгенерированный результат в файл

Шифрование	Дешифрование
------------	--------------

2. Сложный ключ, небольшой текст.

Ключ: 0110101010101101010111011010110110

Исходный текст: 0011000100110100

Шифротекст: 0101101110011001

Шифрование:

Состояние регистра (36 бита)	Сгенерированный ключ
0110101010101101010111011010110110	0110101010101101010111011010110110
Длина введенных состояний: 36	Длина сгенерированного ключа: 36

Исходный текст	Сгенерированный результат
0011000100110100	0101101110011001
Ввести исходный текст из файл	Сохранить сгенерированный результат в файл

Шифрование	Дешифрование
------------	--------------

Дешифрование:

Состояние регистра (36 бита)	Сгенерированный ключ
0110101010101101010111011010110110	0110101010101101010111011010110110
Длина введенных состояний: 36	Длина сгенерированного ключа: 36

Исходный текст	Сгенерированный результат
0101101110011001	0011000100110100
Ввести исходный текст из файл	Сохранить сгенерированный результат в файл

Шифрование	Дешифрование
------------	--------------

3. Простой ключ, большой текст.

Ключ: 1111111111111111111111111111111111

Исходный текст:

1101000010110001110100001011110110100001011101110100011000110011010001100010001101000010
111110110100001011100100100000

Шифротекст:

001011110100111000101111010000010010010111101110100001001101111111100010101110111110000110
101111110000011010100111010000

Шифрование:

Состояние регистра (36 бита)		Сгенерированный ключ	
<pre>111111111111111111111111111111111111</pre>		<pre>11111111111111111111111111111111000000000001111111111000000 000001110000000011111111110001111111000000111110001110</pre>	
Длина введённых состояний: 36		Длина сгенерированного ключа: 120	
Исходный текст		Сгенерированный результат	
<pre>110100001011000111010000101111011011100010000011100110001100100 01110111011010101011101111010010001011001101010110110000</pre>		<pre>0010111101001110001011110100000100101100010000000011001110100100 01110000011010100100010000110001110100001101101000111110</pre>	
Ввести исходный текст из файл		Сохранить сгенерированный результат в файл	
Шифрование		Дешифрование	

Дешифрирование:

Состояние регистра (36 бита)		Сгенерированный ключ	
111111111111111111111111111111111111		11111111111111111111111111111111000000000001111111111000000 000001110000000011111111110001111111000000111110001110	
Длина введенных состояний: 36		Длина сгенерированного ключа: 120	
Исходный текст		Сгенерированный результат	
0010111010011100010111101000010010110001000000011001110100100 01110000011010100100010000110001110100001101101000111110		1101000010110001110100001011111011011100010000011100110001100100 01110111011010101011101111010010001011001101010110110000	
Ввести исходный текст из файл		Сохранить сгенерированный результат в файл	
Шифрование		Дешифрование	

4. Сложный ключ, большой текст.

Ключ: 011010101010110101011101101011010110

Исходный текст:

110100001011000111010000101111101101000010111011110100011000110011010001100010001101000010
111110110100001011100100100000

Шифротекст:

101110100001110010001101000100111011010001110000101101010101000010010111010101001001101101
111100100110110111111110011110

Шифрование:

Состояние регистра (36 бита)	Сгенерированный ключ
011010101010110101011101101011010110	0110101010101101010111011010110101100011000001101011010100001100 01110111101111101001110010000011010101110001000100001011
Длина введенных состояний: 36	Длина сгенерированного ключа: 120
Исходный текст	Сгенерированный результат
110100001011000111010000101111011011100010000011100110001100100 01110111011010101011101111010010001011001101010110110000	101110100001110010001101000100111011111010001110111100101101000 0000000011010100001001110101000101110111100010010111011
Ввести исходный текст из файл	Сохранить сгенерированный результат в файл
Шифрование	Дешифрование

Дешифрование:

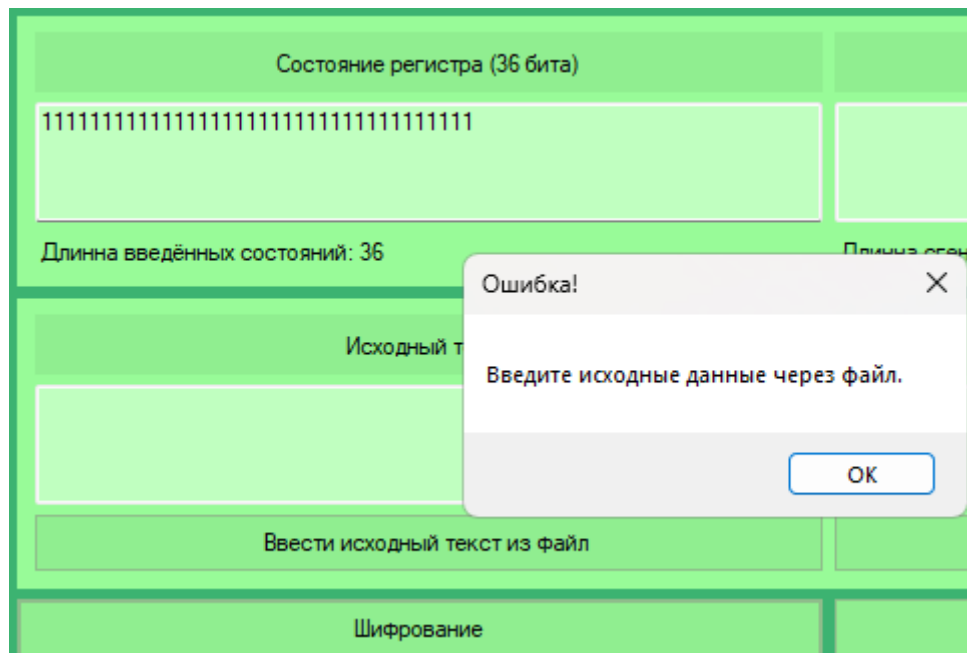
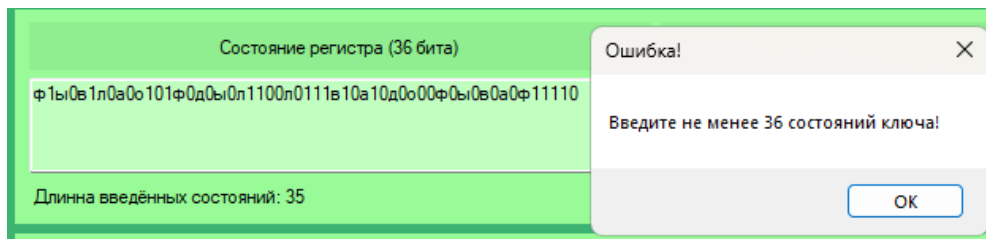
LFSR

Состояние регистра (36 бита)	Сгенерированный ключ
011010101010110101011101101011010110	0110101010101101010111011010110101100011000001101011010100001100 01110111101111101001110010000011010101110001000100001011
Длина введенных состояний: 36	Длина сгенерированного ключа: 120
Исходный текст	Сгенерированный результат
101110100001110010001101000100111011111010001110111100101101000 00000000110101000010011101010001011110111100010010111011	110100001011000111010000101111011011100010000011100110001100100 01110111011010101011101111010010001011001101010110110000
Ввести исходный текст из файл	Сохранить сгенерированный результат в файл
Шифрование	Дешифрование

5. Проверка на не валидных данных.

Состояние регистра (36 бита)	Ошибка!
	Введите исходное состояние для генерации ключа!
Длина введенных состояний: 0	OK

Состояние регистра (36 бита)	Ошибка!
фывлаофдыллвадофываф	Введите не менее 36 состояний ключа!
Длина введенных состояний: 0	OK



Исходный код

```
using System;
using System.IO;
using System.Linq;
using System.Security.Cryptography;
using System.Text;
using System.Web;
using System.Windows.Forms;

namespace ToI_2
{
    public partial class MainForm : Form
    {
        private char[] goodKeys = { '0', '1' };

        private string PLAIN_TEXT;
        private string GENERATED_KEY;
        private string CIPER_TEXT;

        public MainForm()
        {
            InitializeComponent();
            setCurrentKeyLength(0);
        }

        private void setCurrentKeyLength(int length)
        {
            keyLengthLabel.Text = $"Длина введенных состояний: {length}";
        }
    }
}
```

```

private string getCurrentKey(string str)
{
    string res = "";
    for (int i = 0; i < str.Length; i++)
        if (goodKeys.Contains(str[i]))
            res += str[i];
    return res;
}
private (bool, string) getCurrentKeyString(string str)
{
    if (str.Length == 0)
        return (false, "Введите исходное состояние для генерации ключа!");
    string res = getCurrentKey(str);
    if (res.Length < 36)
        return (false, "Введите не менее 36 состояний ключа!");
    return (true, res);
}

private string getFirstBytes(string str, int bytes)
{
    return str.Substring(0, bytes * 8);
}

private string getLastBytes(string str, int bytes)
{
    return str.Substring(str.Length - bytes * 8, bytes * 8);
}

private void setGeneratedText(TextBox textBox, string str)
{
    const int bytesToSee = 8;
    if (str.Length > bytesToSee * 8 * 2)
        textBox.Text = $"Первые 8 байт: {getFirstBytes(str, bytesToSee)}\n\nПоследние 8 байт: {getLastBytes(str, bytesToSee)}";
    else
        textBox.Text = str;
}

private char[] shiftKey(char[] key)
{
    for (int i = 0; i < key.Length - 1; i++)
        key[i] = key[i + 1];
    return key;
}
private char xorNums(char bit1, char bit2)
{
    int num1 = bit1 - '0';
    int num2 = bit2 - '0';
    int result = num1 ^ num2;
    char xorResult = (char)(result + '0');
    return xorResult;
}
private char getNewValue(int[] genPos, char[] key)
{
    char first;
    char second = key[genPos[0] - 1];
    for (var i = 1; i < genPos.Length; i++)
    {
        first = second;
        second = key[genPos[i] - 1];
        second = xorNums(first, second);
    }
}

```



```

        return second;
    }
    private string generateByLFSR(string register, int length)
    {
        char[] genKey = register.ToArray();
        char[] currentKey = register.ToArray();
        int[] xorPositions = { 36, 11 }; // from 1 .. 36

        StringBuilder bits = new StringBuilder();
        for (var i = 0; i < currentKey.Length; i++)
        {
            bits.Append(currentKey[i]);
        }

        while (bits.Length < length)
        {
            genKey = shiftKey(genKey);
            genKey[genKey.Length - 1] = getNewValue(xorPositions, genKey);
            bits.Append(genKey[genKey.Length - 1]);
        }

        return bits.ToString();
    }

    private void keyTextBox_TextChanged(object sender, EventArgs e)
    {
        setCurrentKeyLength(getCurrentKey(keyTextBox.Text).Length);
    }

    private void genKeyButton_Click(object sender, EventArgs e)
    {
        //string str;
        //bool isCorrect;
        //(isCorrect, str) = getCurrentKeyString(keyTextBox.Text);
        //if (!isCorrect)
        //{
        //    MessageBox.Show(str, "Ошибка!");
        //    return;
        //}
        // generate
        //string key = generateByLFSR(str.Substring(0, 36));
        //setGeneratedKey(key);
    }

    private string encryptionAlghoritm(string plain, string key)
    {
        int test = key.Length;
        StringBuilder bits = new StringBuilder();
        for (int i = 0; i < plain.Length; i++)
        {
            char p = plain[i];
            char k = key[i];
            bits.Append(xorNums(p, k));
        }
        return bits.ToString();
    }

    private void ENCRPTION()
    {
        /* Key part */
        string key;
        bool isCorrect;
    }

```

```

        (isCorrect, key) = getCurrentKeyString(keyTextBox.Text);
        if (!isCorrect)
        {
            MessageBox.Show(key, "Ошибка!");
            return;
        }

        /* Plain text part */
        if (PLAIN_TEXT == null || PLAIN_TEXT.Length == 0)
        {
            MessageBox.Show("Введите исходные данные через файл.", "Ошибка!");
            return;
        }

        /* Generate key part */
        GENERATED_KEY = generateByLFSR(key.Substring(0, 36), PLAIN_TEXT.Length);
        setGeneratedText(genKeyTextBox, GENERATED_KEY);

        /* Cipher part */
        CIPHER_TEXT = encryptionAlgoritm(PLAIN_TEXT, GENERATED_KEY);
        setGeneratedText(cypherTextBox, CIPHER_TEXT);

        /* Enables */
        inFileButton.Enabled = true;
        genKeySizeLabel.Text = $"Длина сгенерированного ключа: {GENERATED_KEY.Length}";
    }
    private void button1_Click(object sender, EventArgs e)
    {
        ENCRPTION();
    }

    private (bool, string) ReadFileAsBits(string filePath)
    {
        try
        {
            StringBuilder bits = new StringBuilder();

            using (FileStream fs = new FileStream(filePath, FileMode.Open,
FileAccess.Read))
            {
                int byteRead;
                while ((byteRead = fs.ReadByte()) != -1)
                {
                    bits.Append(Convert.ToString(byteRead, 2).PadLeft(8, '0'));
                }
            }
            if (bits.Length > 0)
                return (true, bits.ToString());

            return (false, null);
        }
        catch
        {
            return (false, null);
        }
    }

    private (bool, string) WriteFileAsBits(string filePath)
    {
        string bits = CIPHER_TEXT;
        try
        {

```

```

        using (FileStream fs = new FileStream(filePath, FileMode.Create,
FileAccess.Write))
        {
            for (int i = 0; i < bits.Length; i += 8)
            {
                string byteBits = bits.Substring(i, Math.Min(8, bits.Length - i));
                byte byteToWrite = Convert.ToByte(byteBits, 2);
                fs.WriteByte(byteToWrite);
            }

            return (true, null);
        }
        catch
        {
            return (false, null);
        }
    }

    private void fromFileButton_Click(object sender, EventArgs e)
    {
        if (openFileDialog.ShowDialog() == DialogResult.OK)
        {
            bool status;
            string result;
            (status, result) = ReadFileAsBits(openFileDialog.FileName);
            if (status) {
                PLAIN_TEXT = result;
                setGeneratedText(plainTextBox, PLAIN_TEXT);
                return;
            }
            MessageBox.Show("Данные в файле не являются корректными.", "Ошибка!");
        }
    }

    private void inFileButton_Click(object sender, EventArgs e)
    {
        if (saveFileDialog.ShowDialog() == DialogResult.OK)
        {
            string str;
            bool status;
            (status, str) = WriteFileAsBits(saveFileDialog.FileName);
            MessageBox.Show(status ? "Данные успешно записаны в файл." : "Ошибка при
записи в файл.", status ? "Успех" : "Ошибка");
        }
    }

    private void button2_Click(object sender, EventArgs e)
    {
        ENCRPTION();
    }
}

```