УО «Белорусский государственный университет информатики и радиоэлектроники»

Кафедра ПОИТ

Отчет по лабораторной работе №2

по предмету

Теория Информации

Вариант 15

Выполнил:

Егоров. А.С.

Проверила:

Болтак С.В.

Группа 351004

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 ***37***). 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*37 + *x*12 + *x*10 + *x*2 + 1

**Тесты**

1. Легкий ключ, маленький текст. (зашифрованный/расшифрованный)

Изображение выглядит как текст, снимок экрана, дисплей, число

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, электроника, снимок экрана, дисплей

Контент, сгенерированный ИИ, может содержать ошибки.

1. Сложный ключ, маленький текст. (зашифрованный/расшифрованный)

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, дисплей, число

Контент, сгенерированный ИИ, может содержать ошибки.

1. Легкий ключ, большой текст. (зашифрованный/расшифрованный)

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

1. Сложный ключ, большой текст. (зашифрованный/расшифрованный)

Изображение выглядит как текст, снимок экрана, Шрифт, дисплей

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

1. Некорректные символы в начальном состоянии LFSR (до/после)

Изображение выглядит как текст, снимок экрана, дисплей, число

Контент, сгенерированный ИИ, может содержать ошибки. Изображение выглядит как текст, снимок экрана, дисплей, число

Контент, сгенерированный ИИ, может содержать ошибки.

**Исходный код**

using System.Text;

namespace Lab2;

public partial class Main : Form

{

private const int RegLen = 37;

// x37 + x12 + x10 + x2 + 1

private readonly byte[] \_bits = [2,10,12,37];

public Main()

{

InitializeComponent();

}

private void btn\_crypto\_Click(object sender, EventArgs e)

{

tb\_lfsr.Text = CleanText(tb\_lfsr.Text);

string seed = tb\_lfsr.Text;

if (!ValidateSeed(seed)) return;

tb\_sourceText.Text = CleanText(tb\_sourceText.Text);

string sourceText = tb\_sourceText.Text;

string key = GetKey(seed, sourceText.Length);

// вывод полученного ключа

tb\_key.Text = key;

string cryptoText = Xor(sourceText, key);

// вывод зашифрованного текста в бинарном виде (8 бит на символ)

tb\_cryptoText.Text = cryptoText;

}

private void btn\_openFile\_Click(object sender, EventArgs e)

{

if (openFileDialog1.ShowDialog() != DialogResult.OK) return;

string data = File.ReadAllText(openFileDialog1.FileName);

tb\_sourceText.Text = CleanText(data);

}

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

{

if (saveFileDialog1.ShowDialog() != DialogResult.OK) return;

File.WriteAllText(saveFileDialog1.FileName, tb\_cryptoText.Text);

MessageBox.Show($"Файл сохранён как {saveFileDialog1.FileName}");

}

private bool ValidateSeed(string seed)

{

if (seed.Length == RegLen) return true;

MessageBox.Show("Размер начального состояния должен быть 37 символов!");

return false;

}

private string GetKey(string seed, int lenght)

{

byte[] curState = seed.Select(bit => (byte)(bit - '0')).ToArray();

StringBuilder sb = new StringBuilder(lenght);

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

{

// добавляем старший бит в результат

sb.Append(curState[0]);

// получаем бит, которые будет задвигать в начало

byte newBit = 0;

foreach (var bit in \_bits)

{

newBit ^= curState[^bit];

}

// сдвигаем все кроме первого бит

Array.Copy(curState, 1, curState, 0, curState.Length - 1);

// сдвигаем последний бит

curState[^1] = newBit;

}

return sb.ToString();

}

private string Xor(string data, string key)

{

// инициализируем результат

StringBuilder sb = new StringBuilder(data.Length);

// оформляем xor

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

{

if ((data[i] - '0' + key[i] - '0') == 1)

sb.Append('1');

else

sb.Append('0');

}

// выводим результат

return sb.ToString();

}

private string CleanText(string text)

{

return string.Concat(text.Where(c => c == '0' || c == '1'));

}

}