

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ**

**ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

**(ДГТУ)**

Факультет «Информатика и вычислительная техника»

Кафедра «Кибербезопасность информационных систем»

Лабораторная работа №8

по дисциплине: «Методы и средства криптографической защиты информации»

На тему «РСЛОС и скремблирование»

Выполнил обучающийся гр. ВКБ42

Михайлов Александр Сергеевич

          Проверила:

Сафарьян О.А

Ростов-на-Дону

2024

Цель работы: формирование умений шифрования с использованием регистра сдвигов с линейной обратной связью для получения периода последовательности и скремблирования.

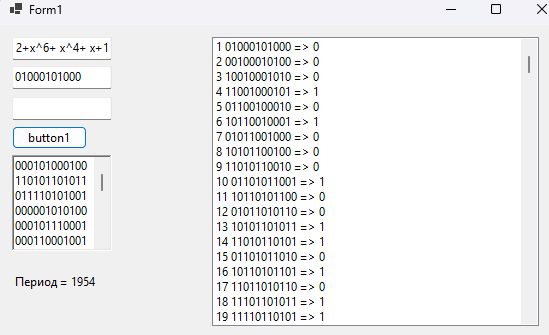
Задание 1:

Использование РСЛОС для получения периода последовательности

Значение: 01000101000

Многочлен: x^12+x^6+ x^4+ x+1

Программная реализация:



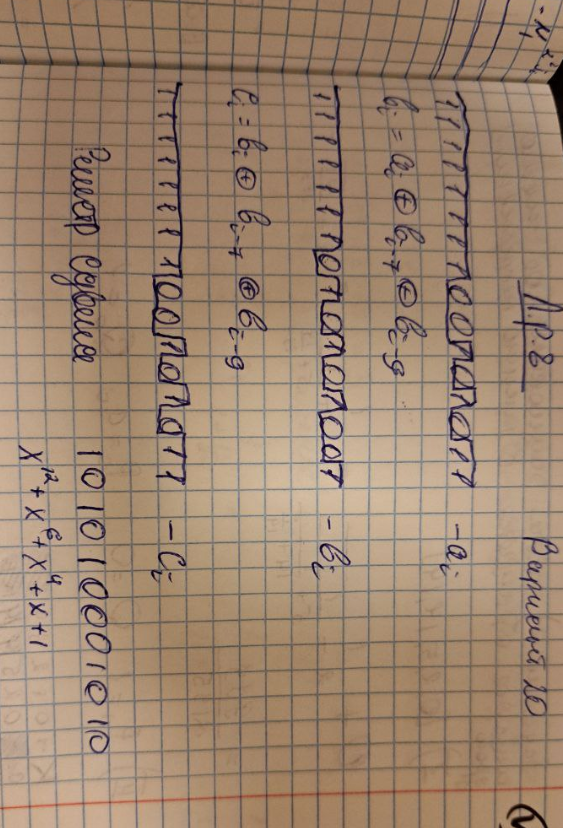
Задание 2:

Выполнить скремблирование.

Значение: 11111111100101011

Правило: bi=ai^ bi-7 ^ bi-9

Ручная реализация:



Вывод:

В ходе выполнения лабораторной работы были сформированы умений шифрования с использованием регистра сдвигов с линейной обратной связью и скремблирования.

Листинг кода

using System.Diagnostics;

using System.Text.RegularExpressions;

namespace РСЛОС

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

int len;

bool hasLen = int.TryParse(textBox3.Text, out len);

string polynom = textBox1.Text.Trim();

string binary = textBox2.Text;

int count = 1;

string posl = "";

int period = 0;

if (string.IsNullOrWhiteSpace(polynom))

{

MessageBox.Show("Введите многочлен!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

string OriginalBinary = binary;

var (n, digits) = ParsePolynomial(polynom);

listBox1.Items.Clear();

int temp = 0;

string reversed = new string(binary.Reverse().ToArray());

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

{

if (digits.Contains(i))

{

temp += int.Parse(reversed[i].ToString());

}

}

char number = ' ';

if (temp % 2 == 0)

{

number = '0';

}

else

{

number = '1';

}

char lastChar = binary[binary.Length - 1];

listBox1.Items.Add(count + " " + binary + " => " + lastChar);

posl = posl + lastChar;

binary = binary.Remove(binary.Length - 1);

binary = number + binary;

while ((!hasLen && binary != OriginalBinary) || (hasLen && count < len))

{

count++;

temp = 0;

reversed = new string(binary.Reverse().ToArray());

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

{

if (digits.Contains(i))

{

temp += int.Parse(reversed[i].ToString());

}

}

number = (temp % 2 == 0) ? '0' : '1';

lastChar = binary[binary.Length - 1];

listBox1.Items.Add(count + " " + binary + " => " + lastChar);

posl = posl + lastChar;

binary = binary.Remove(binary.Length - 1);

binary = number + binary;

}

if (!hasLen)

{

reversed = new string(binary.Reverse().ToArray());

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

{

if (digits.Contains(i))

{

temp += int.Parse(reversed[i].ToString());

}

}

number = (temp % 2 == 0) ? '0' : '1';

lastChar = binary[binary.Length - 1];

listBox1.Items.Add("Начало повторения " + binary + " => " + lastChar);

binary = binary.Remove(binary.Length - 1);

binary = number + binary;

}

period = count;

label2.Text = "Период = " + period;

Debug.WriteLine(posl.Length);

richTextBox1.Text = posl;

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка обработки: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private (int n, List<int> digits) ParsePolynomial(string polynomial)

{

Regex termRegex = new Regex(@"x**\^**(\d+)|x|(\d+)");

var matches = termRegex.Matches(polynomial);

List<int> powers = new List<int>();

foreach (Match match in matches)

{

if (match.Groups[1].Success)

{

powers.Add(int.Parse(match.Groups[1].Value));

}

else if (match.Value == "x")

{

powers.Add(1);

}

else if (match.Value == "1")

{

powers.Add(0);

}

}

if (!powers.Any())

throw new Exception("Некорректный многочлен!");

int n = powers.Max();

powers.Remove(n);

return (n, powers);

}