ГУАП

КАФЕДРА № 33

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

ассистент		А.И. Дубинин
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ

РЕАЛИЗАЦИЯ МЕТОДОВ ДЕШИФРОВАНИЯ ЛОГ-ЖУРНАЛОВ, СФОРМИРОВАННЫХ ЗАЩИЩЕННЫМ КЛИЕНТСКИМ ПРИЛОЖЕНИЕМ

по курсу: ТЕХНОЛОГИИ И МЕТОДЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 3135		Д.Д. Лобко
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2023

СОДЕРЖАНИЕ

Цель работы:

Получить навыки работы с криптографическими алгоритмами в среде программирования Microsoft Visual Studio 2022. Реализовать декодирование лог-журналов защищенного клиентского приложения с использованием Windows Forms на языке C#.

Основное задание:

1. Разработать функционал формирования декодирования лог-журнала обработки исключений, который был подвергнут шифрованию в соответствии с заданным вариантом.
2. Форма ввода ключа шифрования должна быть модальной, а сам ключ формироваться в соответствии с правилами «сильных ключей»
3. Оформить отчет по лабораторной работе, содержащий: титульный лист, название, номер и цель работы, постановку задачи, алгоритм решения для каждого программного модуля, листинг программных модулей, распечатку результатов, распечатку изображения форм, используемых в программе.

Вариант № 11

Симметричный метод шифрования. Алгоритм гаммирования (побитовое XOR)

Для реализации шифрования лог-журнала гаммированием с помощью операции XOR, необходимо формировать ключ в соответствии с правилами «сильных ключей». Это реализовано в Microsoft .NET Framework, с помощью класса “Guid”. Используя методы этого класса, можно сгенерировать строку символов, которая будет являться «сильным ключом» и будет использоваться для шифрования лог-журнала. Также необходимо оставить возможность пользователю ключ самостоятельно. Поэтому ключ будет генерироваться в элемент “textBox”, где его можно будет редактировать.

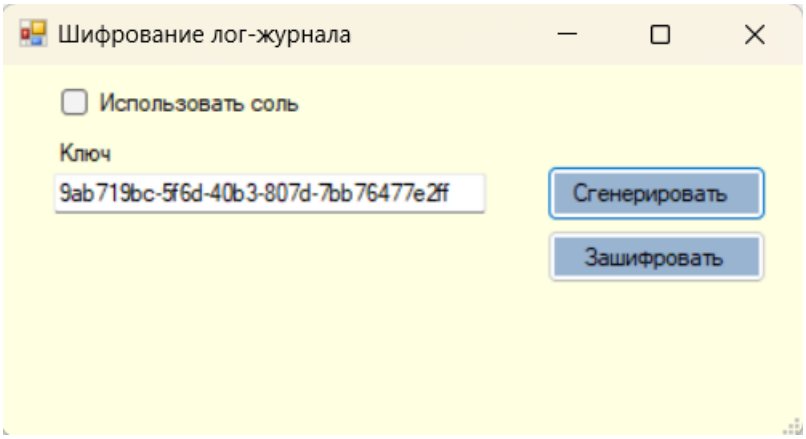

Рис. 1 Интерфейс формы для шифрования лог-журнала с генерацией ключа

Шифрование будет происходит следующим образом: над каждым символом лог-журнала, приведенному к типу “byte”, и каждым символом ключа, приведенному к тому же типу, будет производится операция логического исключающего ИЛИ (XOR), и

присваиваться массиву из байт. Затем этот массив будет записываться в бинарный файл расширения “.xrd”, и затем выводиться в поле для логов. Аналогично будет происходить расшифровка, только первые символы для операции XOR будет браться из раннее созданного файла. Когда для шифрования длина ключа заканчивается, символы ключа для шифрования будут браться с начала (с нулевого индекса).

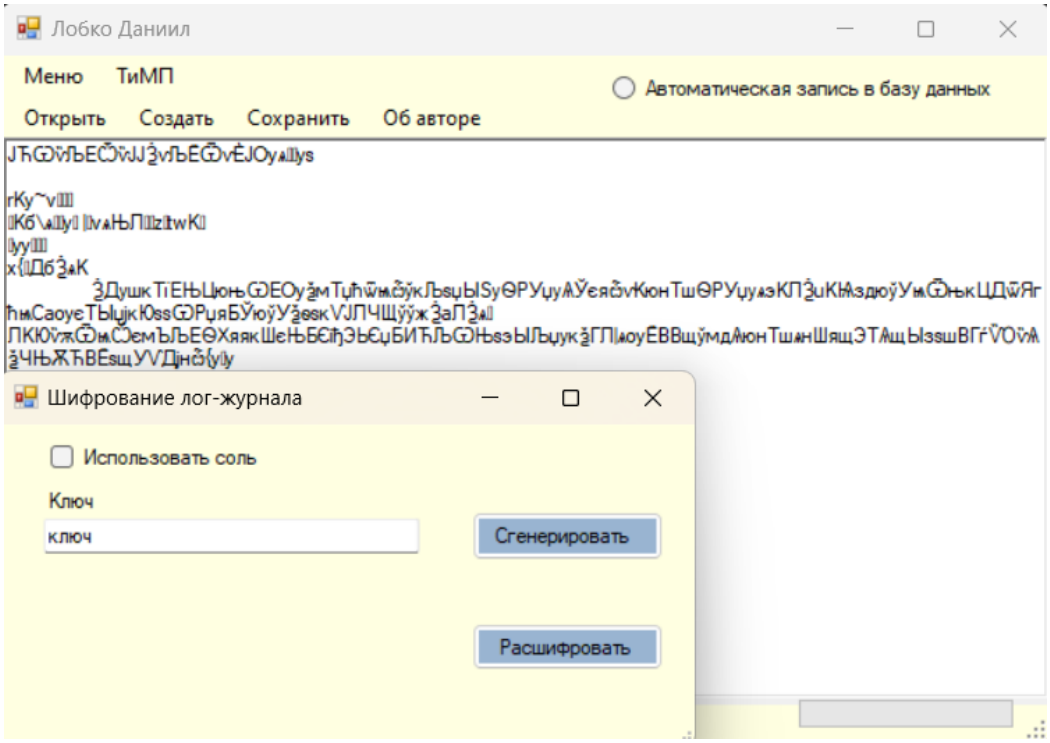

Рис. 2 Пример шифрования лог-журнала

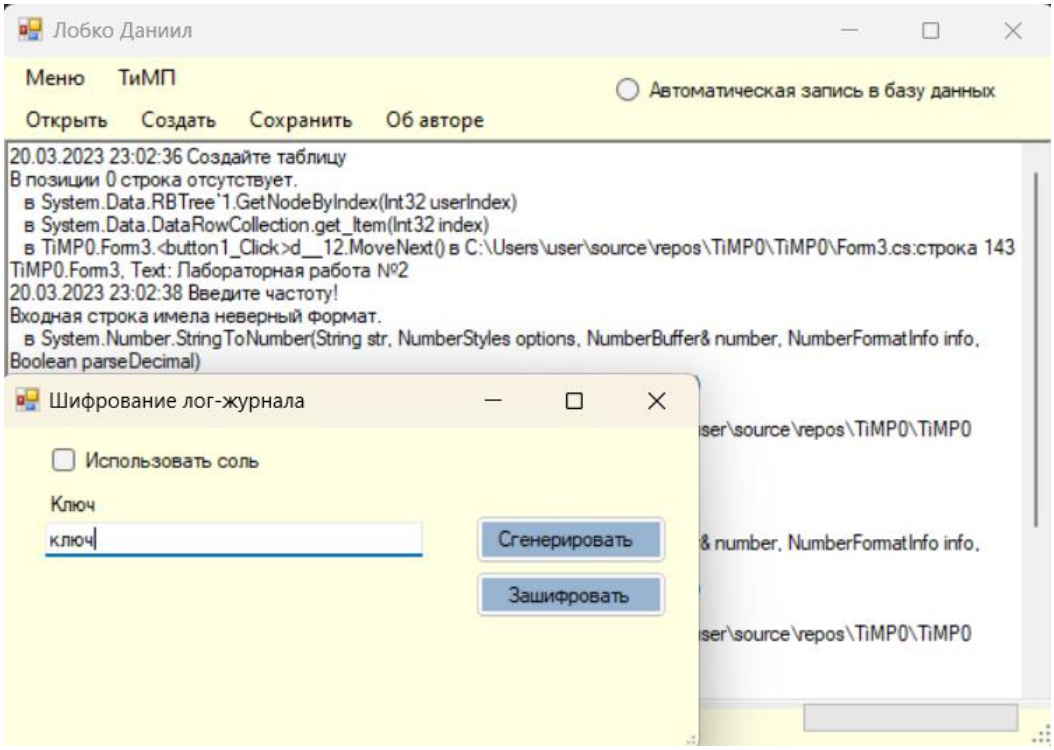

Рис. 3 Пример расшифрования лог-журнала

В качестве дополнительного задания можно усложнить шифрование путем добавления к ключу соли. Соль не будет генерироваться, но пользователь сможет ввести ее сам.

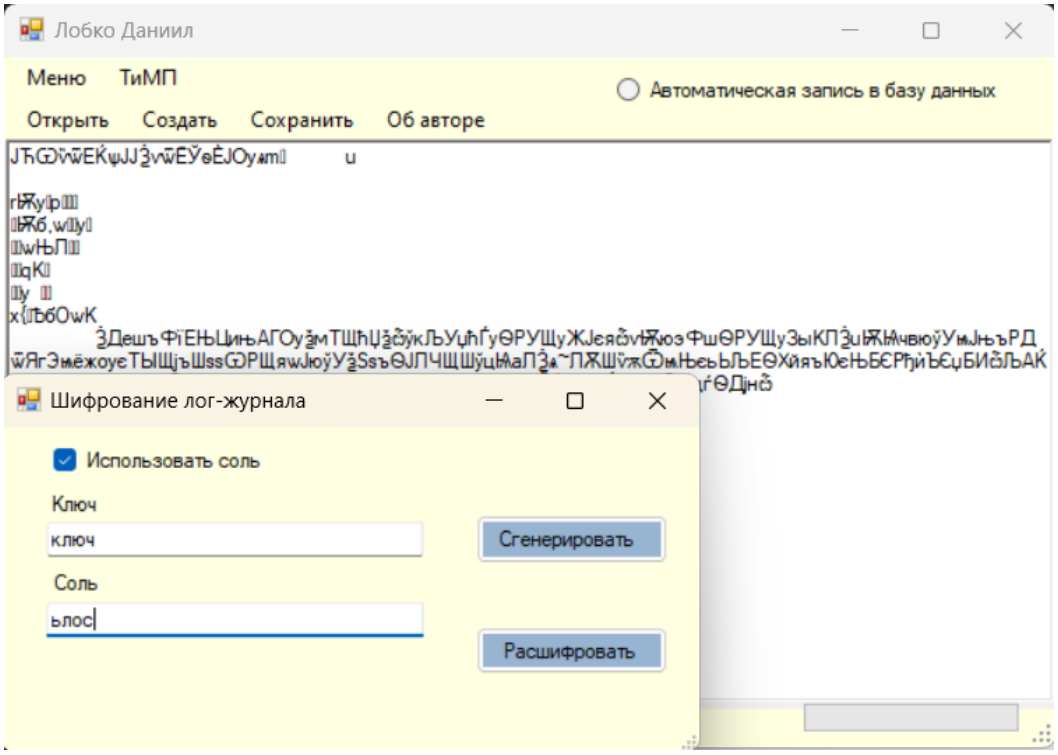

Рис. 4 Пример шифрования лог-журнала с солью

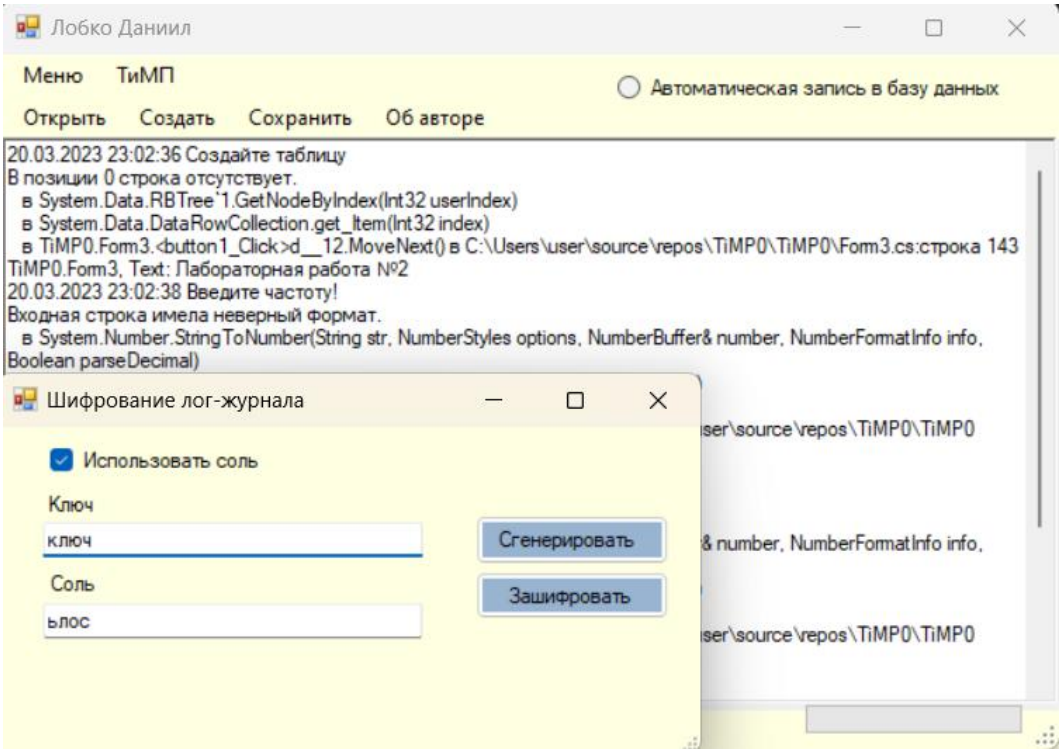

Рис. 5 Пример расшифровки с применением соли

В приведенных скриншотах можно увидеть различия между шифрованием с солью и без с помощью одного и того же ключа.

Вывод

В данной лабораторной работе было реализовано декодирование лог-журнала защищенного клиентского приложения с помощью логической операции XOR и получены навыки работы с криптографическими алгоритмами в среде програмиирования Microsoft Visual Studio.

ПРИЛОЖЕНИЕ

Листинг 1. Код формы с шифрованием

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static TiMP0.Form1;

namespace TiMP0
{
    public partial class Form9 : Form
    {
        public Form9()
        {
            InitializeComponent();
        }
        public RichTextBox logs;
        public RadioButton dbcheck;

        private string GetRepeatKey(string s, int n)

        {
            int r = 1;
            char[] chars = new char[n];
            for (int i =0; i<n; i++)
            {
                r = ((r * 1103515247) / 65536) % 256;
                chars[i] = (char)r;
            }
            s = new string(chars);
            return s;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            var guid = Guid.NewGuid();
            textBox1.Text = guid.ToString();
        }

        async private void button2_Click(object sender, EventArgs e)
        {
            if (checkBox1.Checked == false)
            {
                try
                {
                    byte[] gammaBytes = Encoding.Unicode.GetBytes(textBox1.Text);
                    byte[] logBytes = Encoding.Unicode.GetBytes(logs.Text);
                    /*if (File.Exists("log.xrd"))
                        logBytes = File.ReadAllBytes("log.xrd");*/
                    byte[] cipheredBytes = new byte[logBytes.Length];

                    for (int i = 0; i<logBytes.Length; i++)
                        cipheredBytes[i] = (byte)(logBytes[i] ^ gammaBytes[i %
gammaBytes.Length]);

                    File.WriteAllBytes("log.xrd", cipheredBytes);
                    logs.Text = Encoding.Unicode.GetString(cipheredBytes);
                    button2.Visible = false; button3.Visible = true;
                 }
```

```
                catch (Exception ex)
                {
                    string date = DateTime.Now.ToString();
                    //await Task.prgbar.PerformStep();
                    string indexf = Form.ActiveForm.ToString();
                    indexf = indexf.Split(',')[0];
                    logs.Text += DateTime.Now + $" Сгенерируйте ключ!" + "\r\n" +
ex.Message + "\r\n" + ex.StackTrace + "\r\n" + Form.ActiveForm + "\r\n";
                    if (dbcheck.Checked == true)
                    {
                        await MyPGSQLSender.Main(ex.Message, ex.StackTrace, date,
indexf);
                    }
                }
            }
            else if(checkBox1.Checked == true)
            {
                try
                {
                    string hash = textBox1.Text;
                    hash += textBox2.Text;
                    byte[] gammaBytes = Encoding.Unicode.GetBytes(hash);
                    byte[] logBytes = Encoding.Unicode.GetBytes(logs.Text);
                    /*if (File.Exists("log.xrd"))
                        logBytes = File.ReadAllBytes("log.xrd");*/
                    byte[] cipheredBytes = new byte[logBytes.Length];

                    for (int i = 0; i < logBytes.Length; i++)
                        cipheredBytes[i] = (byte)(logBytes[i] ^ gammaBytes[i %
gammaBytes.Length]);

                    File.WriteAllBytes("log.xrd", cipheredBytes);
                    logs.Text = Encoding.Unicode.GetString(cipheredBytes);
                    button2.Visible = false; button3.Visible = true;
                }
                catch (Exception ex)
                {
                    string date = DateTime.Now.ToString();
                    //await Task.prgbar.PerformStep();
                    string indexf = Form.ActiveForm.ToString();
                    indexf = indexf.Split(',')[0];
                    logs.Text += DateTime.Now + $" Сгенерируйте ключ и введите
соль!" + "\r\n" + ex.Message + "\r\n" + ex.StackTrace + "\r\n" + Form.ActiveForm +
"\r\n";
                    if (dbcheck.Checked == true)
                    {
                        await MyPGSQLSender.Main(ex.Message, ex.StackTrace, date,
indexf);
                    }
                }
            }
        }

        private void radioButton1_CheckedChanged(object sender, EventArgs e)
        {

        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            if (checkBox1.Checked == true)
            {
                label2.Visible = true;
                textBox2.Visible = true;
            }
           else if(checkBox1.Checked == false)
           {
```

```
                label2.Visible = false;
                textBox2.Visible = false;
            }

        }

        async private void button3_Click(object sender, EventArgs e)
        {
            if (checkBox1.Checked == false)
            {
                try
                {
                    byte[] gammaBytes = Encoding.Unicode.GetBytes(textBox1.Text);
                    byte[] logBytes = Encoding.Unicode.GetBytes(logs.Text);
                    if (File.Exists("log.xrd"))
                        logBytes = File.ReadAllBytes("log.xrd");
                    byte[] cipheredBytes = new byte[logBytes.Length];

                    for (int i = 0; i < logBytes.Length; i++)
                        cipheredBytes[i] = (byte)(logBytes[i] ^ gammaBytes[i %
gammaBytes.Length]);
                    File.Delete("log.xrd");
                    //File.WriteAllBytes("log.xrd", cipheredBytes);
                    logs.Text = Encoding.Unicode.GetString(cipheredBytes);
                    button3.Visible = false; button2.Visible = true;
                }
                catch (Exception ex)
                {
                    string date = DateTime.Now.ToString();
                    //await Task.prgbar.PerformStep();
                    string indexf = Form.ActiveForm.ToString();
                    indexf = indexf.Split(',')[0];
                    logs.Text += DateTime.Now + $" Сгенерируйте ключ!" + "\r\n" +
ex.Message + "\r\n" + ex.StackTrace + "\r\n" + Form.ActiveForm + "\r\n";
                    if (dbcheck.Checked == true)
                        await MyPGSQLSender.Main(ex.Message, ex.StackTrace, date,
indexf);
                }
            } else if (checkBox1.Checked == true)
            {
                try
                {
                    string hash = textBox1.Text;
                    hash += textBox2.Text;
                    byte[] gammaBytes = Encoding.Unicode.GetBytes(hash);
                    byte[] logBytes = Encoding.Unicode.GetBytes(logs.Text);
                    if (File.Exists("log.xrd"))
                        logBytes = File.ReadAllBytes("log.xrd");
                    byte[] cipheredBytes = new byte[logBytes.Length];

                    for (int i = 0; i < logBytes.Length; i++)
                        cipheredBytes[i] = (byte)(logBytes[i] ^ gammaBytes[i %
gammaBytes.Length]);
                    File.Delete("log.xrd");
                    //File.WriteAllBytes("log.xrd", cipheredBytes);
                    logs.Text = Encoding.Unicode.GetString(cipheredBytes);
                    button3.Visible = false; button2.Visible = true;
                }
                catch (Exception ex)
                {
                    string date = DateTime.Now.ToString();
                    //await Task.prgbar.PerformStep();
                    string indexf = Form.ActiveForm.ToString();
                    indexf = indexf.Split(',')[0];
                    logs.Text += DateTime.Now + $" Сгенерируйте ключ и введите
соль!" + "\r\n" + ex.Message + "\r\n" + ex.StackTrace + "\r\n" + Form.ActiveForm +
"\r\n";
```

```
                    if (dbcheck.Checked == true)
                        await MyPGSQLSender.Main(ex.Message, ex.StackTrace, date,
indexf);
                }
            }
        }
    }

}
```