

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 №1

по дисциплине: «Методы и средства криптографической защиты информации»

На тему «Шифрование с использованием метода шифрующих таблиц и метода магического квадрата»

Выполнил обучающийся гр. ВКБ42

Михайлов Александр Сергеевич

          Проверила:

Сафарьян О.А

Ростов-на-Дону

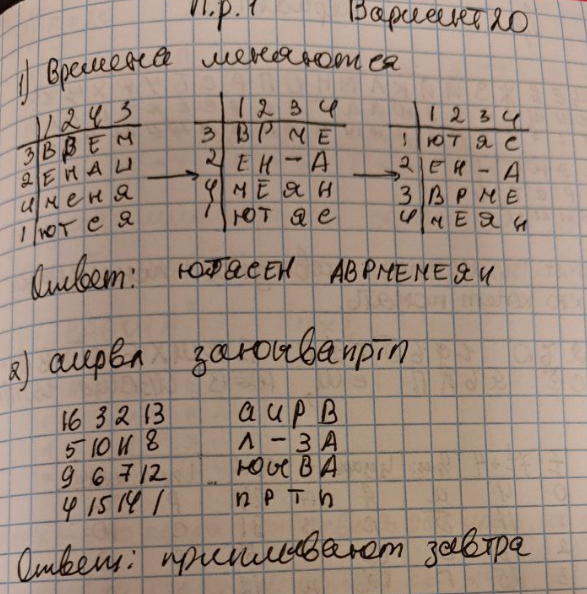
2024

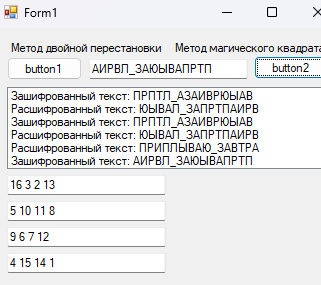
Цель работы: формирование умений шифрования с использованием методов шифрующих таблиц и магического квадрата.

Задание 1 Вариант 20

Выполните шифрование/расшифрование, используя метод шифрующих таблиц согласно варианту. При шифровании/расшифровании все пробелы учитываются. Пробелы в шифр тексте обозначаются символом «\_», несколько подряд идущих пробе- лов разделяются символом «|».

Ручная отработка:



Программная реализация:  


Листинг кода:

namespace Cripto\_1

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

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

{

// Получаем значение из TextBox1

string inputText = textBox1.Text;

// Удаляем пробелы

inputText = inputText.Replace(" ", "");

// Проверяем, что длина текста достаточна для заполнения таблицы 4x4

if (inputText.Length < 16)

{

MessageBox.Show("Текст должен содержать не менее 16 символов без учета пробелов.");

return;

}

// Создаем таблицу 4x4 и заполняем ее символами из inputText

char[,] table = new char[4, 4];

int index = 0;

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

if (index < inputText.Length)

{

table[i, j] = inputText[index++];

}

else

{

table[i, j] = ' '; // Заполняем оставшиеся ячейки пробелами

}

}

}

// Последовательности номеров столбцов и строк для перестановки

int[] columnOrder = { 1, 2, 4, 3 };

int[] rowOrder = { 4, 2, 1, 3 };

// Создаем новую таблицу для результата перестановки столбцов

char[,] permutedColumnsTable = new char[4, 4];

// Выполняем перестановку столбцов

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

permutedColumnsTable[i, j] = table[i, columnOrder[j] - 1];

}

}

// Создаем новую таблицу для результата перестановки строк

char[,] finalTable = new char[4, 4];

// Выполняем перестановку строк

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

finalTable[i, j] = permutedColumnsTable[rowOrder[i] - 1, j];

}

}

// Выводим результат перестановки в ListBox1

string result = "";

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

result += finalTable[i, j];

}

}

listBox1.Items.Add("Зашифрованный текст: " + result);

// Расшифровка

// Создаем таблицу для расшифрованного текста

char[,] decryptedTable = new char[4, 4];

// Обратная перестановка строк

int[] reverseRowOrder = { 2, 1, 3, 0 }; // Индексы для обратной перестановки строк

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

decryptedTable[reverseRowOrder[i], j] = finalTable[i, j];

}

}

// Обратная перестановка столбцов

int[] reverseColumnOrder = { 0, 1, 3, 2 }; // Индексы для обратной перестановки столбцов

char[,] finalDecryptedTable = new char[4, 4];

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

finalDecryptedTable[i, reverseColumnOrder[j]] = decryptedTable[i, j];

}

}

// Выводим результат расшифровки в ListBox1

string decryptedResult = "";

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

decryptedResult += finalDecryptedTable[i, j];

}

}

listBox1.Items.Add("Расшифрованный текст: " + decryptedResult);

}

private void button2\_Click(object sender, EventArgs e)

{

// Получаем значение из TextBox1

string encryptedText = textBox1.Text;

// Проверяем, что длина текста достаточна для заполнения таблицы 4x4

if (encryptedText.Length < 16)

{

MessageBox.Show("Текст должен содержать не менее 16 символов.");

return;

}

// Получаем значения из TextBox2, TextBox3, TextBox4 и TextBox5

string[] rows = new string[4];

rows[0] = textBox2.Text;

rows[1] = textBox3.Text;

rows[2] = textBox4.Text;

rows[3] = textBox5.Text;

// Создаем магический квадрат

int[,] magicSquare = new int[4, 4];

// Заполняем магический квадрат значениями из TextBox2, TextBox3, TextBox4 и TextBox5

for (int i = 0; i < 4; i++)

{

// Разделяем строку на числа

string[] values = rows[i].Split(' ');

// Проверяем, что в строке 4 числа

if (values.Length != 4)

{

MessageBox.Show($"Строка {i + 1} должна содержать 4 числа.");

return;

}

// Преобразуем строки в числа и заполняем квадрат

for (int j = 0; j < 4; j++)

{

if (!int.TryParse(values[j], out magicSquare[i, j]))

{

MessageBox.Show($"Значение '{values[j]}' в строке {i + 1} не является числом.");

return;

}

}

}

// Проверка магического квадрата

int magicSum = 34; // Сумма элементов в магическом квадрате 4x4 должна быть 34

bool isMagicSquare = true;

// Проверка сумм по горизонтали

for (int i = 0; i < 4; i++)

{

int rowSum = 0;

for (int j = 0; j < 4; j++)

{

rowSum += magicSquare[i, j];

}

if (rowSum != magicSum)

{

isMagicSquare = false;

break;

}

}

// Проверка сумм по вертикали

if (isMagicSquare)

{

for (int j = 0; j < 4; j++)

{

int colSum = 0;

for (int i = 0; i < 4; i++)

{

colSum += magicSquare[i, j];

}

if (colSum != magicSum)

{

isMagicSquare = false;

break;

}

}

}

// Проверка сумм по диагоналям

if (isMagicSquare)

{

int diag1Sum = 0;

int diag2Sum = 0;

for (int i = 0; i < 4; i++)

{

diag1Sum += magicSquare[i, i];

diag2Sum += magicSquare[i, 3 - i];

}

if (diag1Sum != magicSum || diag2Sum != magicSum)

{

isMagicSquare = false;

}

}

// Если квадрат не является магическим, выводим сообщение об ошибке

if (!isMagicSquare)

{

MessageBox.Show("Магический квадрат не является корректным.");

return;

}

// Создаем таблицу 4x4 и заполняем ее символами из encryptedText

char[,] table = new char[4, 4];

int index = 0;

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

table[i, j] = encryptedText[index++];

}

}

// Создаем таблицу для расшифрованного текста

char[,] decryptedTable = new char[4, 4];

// Заполняем decryptedTable в соответствии с магическим квадратом

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

int value = magicSquare[i, j];

int row = (value - 1) / 4;

int col = (value - 1) % 4;

decryptedTable[i, j] = table[row, col];

}

}

// Выводим результат расшифровки в ListBox1

string decryptedResult = "";

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

decryptedResult += decryptedTable[i, j];

}

}

listBox1.Items.Add("Расшифрованный текст: " + decryptedResult);

// Шифрование расшифрованного текста тем же методом

char[,] encryptedTable = new char[4, 4];

// Заполняем encryptedTable в соответствии с магическим квадратом

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

int value = magicSquare[i, j];

int row = (value - 1) / 4;

int col = (value - 1) % 4;

encryptedTable[row, col] = decryptedTable[i, j];

}

}

// Выводим результат шифрования в ListBox1

string encryptedResult = "";

for (int i = 0; i < 4; i++)

{

for (int j = 0; j < 4; j++)

{

encryptedResult += encryptedTable[i, j];

}

}

listBox1.Items.Add("Зашифрованный текст: " + encryptedResult);

}

}

}

Вывод:

В ходе выполнения лабораторной работы были сформированы умения шифрования с использованием методов шифрующих таблиц и магического квадрата