

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 №3

по дисциплине: «Методы и средства криптографической защиты информации»

На тему «Реализация алгоритма Плейфейра»

Выполнил обучающийся гр. ВКБ42

Михайлов Александр Сергеевич

          Проверила:

Сафарьян О.А

Ростов-на-Дону

2024

Цель работы: формирование умений шифрования с использованием алгоритма шифрования Плейфейра.

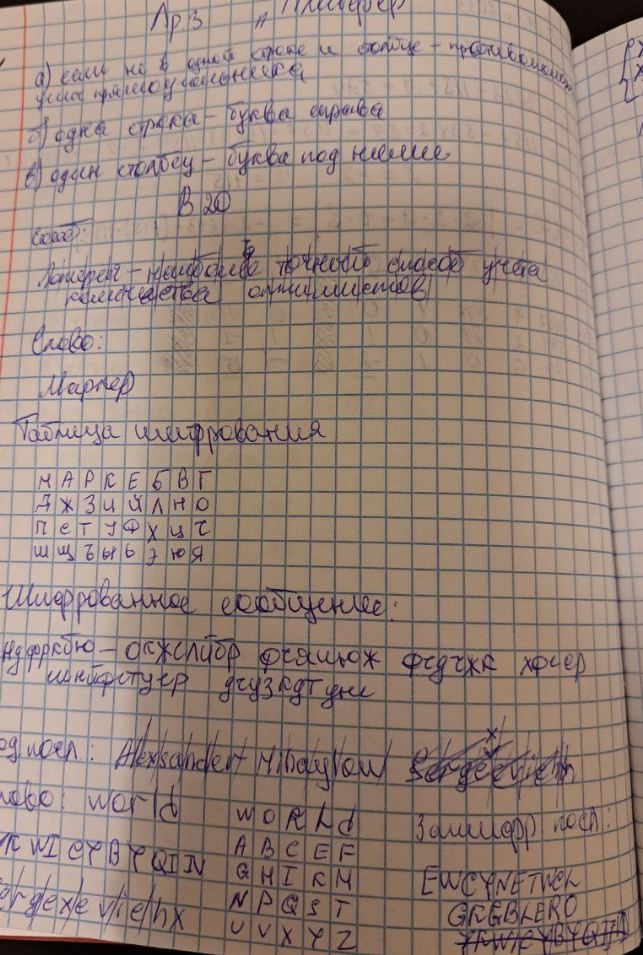
Задание:

Зашифруйте сообщение, используя алгоритм Плейфейра согласно своему варианту. Размер шифрующей таблицы 4 × 8.

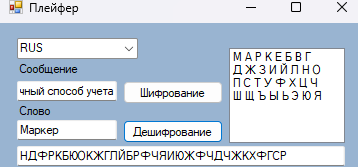
Сообщение: Лотерея – наиболее точный способ учета количества оптимистов

Ключ: Маркер

Ручная реализация:



Программная реализация:



Вывод:

В ходе выполнения лабораторной работы были сформированы умения шифрования с использованием алгоритма шифрования Плейфейра

Листинг кода

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Pleyfer

{

public partial class Form1 : Form

{

private char[,] matrix = new char[4, 8];

private string key;

public bool IsRussian = true;

public string alphabet;

public Form1()

{

InitializeComponent();

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox1.SelectedItem.ToString() == "RUS")

{

IsRussian = true;

matrix = new char[4, 8];

}

else if (comboBox1.SelectedItem.ToString() == "ENG")

{

IsRussian = false;

matrix = new char[5, 5];

}

}

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

{

key = PrepareKey(textBox2.Text);

GenerateMatrix();

string plaintext = PrepareText(textBox1.Text);

textBox3.Text = Encrypt(plaintext);

DisplayMatrixInListBox();

}

private string PrepareKey(string key)

{

if (IsRussian)

{

key = key.ToUpper().Replace("ё", "е");

}

else

{

key = key.ToUpper().Replace("j", "i");

}

key = new string(key.Distinct().ToArray());

return key;

}

private void DisplayMatrixInListBox()

{

int MatrixRows = 0;

int MatrixColumns = 0;

if (IsRussian)

{

MatrixRows = 4;

MatrixColumns = 8;

}

else

{

MatrixRows = 5;

MatrixColumns = 5;

}

listBox1.Items.Clear();

for (int row = 0; row < MatrixRows; row++)

{

string rowString = "";

for (int col = 0; col < MatrixColumns; col++)

{

rowString += matrix[row, col] + " ";

}

listBox1.Items.Add(rowString);

}

}

private void GenerateMatrix()

{

if (IsRussian)

{

GenerateRussianMatrix();

}

else

{

GenerateEnglishMatrix();

}

}

private void GenerateRussianMatrix()

{

alphabet = "АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

var usedChars = new HashSet<char>(key);

int index = 0;

foreach (char c in key)

{

matrix[index / 8, index % 8] = c;

index++;

}

foreach (char c in alphabet)

{

if (!usedChars.Contains(c))

{

matrix[index / 8, index % 8] = c;

index++;

}

}

if (index < 32)

{

matrix[3, 7] = ' ';

}

}

private void GenerateEnglishMatrix()

{

alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ";

var usedChars = new HashSet<char>(key);

int index = 0;

foreach (char c in key)

{

matrix[index / 5, index % 5] = c;

index++;

}

foreach (char c in alphabet)

{

if (!usedChars.Contains(c))

{

matrix[index / 5, index % 5] = c;

index++;

}

}

}

private string PrepareText(string input)

{

if (IsRussian)

{

input = input.ToUpper().Replace("Ё", "Е").Replace(" ", "");

}

else

{

input = input.ToUpper().Replace("J", "I").Replace(" ", "");

}

string output = "";

string filteredInput = "";

foreach (char c in input)

{

if (alphabet.Contains(c))

{

filteredInput += c;

}

}

for (int i = 0; i < filteredInput.Length; i += 2)

{

char firstChar = filteredInput[i];

char secondChar = (i + 1 < filteredInput.Length) ? filteredInput[i + 1] : '\0';

if (secondChar != '\0' && firstChar == secondChar)

{

output += firstChar + (IsRussian ? "Ъ" : "X") + secondChar;

}

else

{

output += firstChar;

if (secondChar != '\0')

{

output += secondChar;

}

else

{

output += (IsRussian ? "Ъ" : "X");

}

}

}

if (output.Length % 2 != 0)

{

output += IsRussian ? "Ъ" : "X";

}

Debug.WriteLine(output);

return output;

}

private string Encrypt(string input)

{

string output = "";

for (int i = 0; i < input.Length; i += 2)

{

var pair = EncryptPair(input[i], input[i + 1]);

output += pair;

}

return output;

}

private string Decrypt(string input)

{

string output = "";

for (int i = 0; i < input.Length; i += 2)

{

Debug.WriteLine(input[i]+ " " + input[i+1]);

var pair = DecryptPair(input[i], input[i + 1]);

Debug.WriteLine(pair);

output += pair;

}

return output;

}

private string EncryptPair(char a, char b)

{

int MatrixRows = 0;

int MatrixColumns = 0;

if (IsRussian)

{

MatrixRows = 4;

MatrixColumns = 8;

}

else

{

MatrixRows = 5;

MatrixColumns = 5;

}

(int rowA, int colA) = FindPosition(a);

(int rowB, int colB) = FindPosition(b);

if (rowA == rowB)

{

return $"{matrix[rowA, ((colA + 1) % MatrixColumns + MatrixColumns) % MatrixColumns]}{matrix[rowB, ((colB + 1) % MatrixColumns + MatrixColumns) % MatrixColumns]}";

}

else if (colA == colB)

{

return $"{matrix[((rowA + 1) % MatrixRows + MatrixRows) % MatrixRows, colA]}{matrix[((rowB + 1) % MatrixRows + MatrixRows) % MatrixRows, colB]}";

}

else

{

return $"{matrix[rowA, colB]}{matrix[rowB, colA]}";

}

}

private string DecryptPair(char a, char b)

{

int MatrixRows = 0;

int MatrixColumns = 0;

if (IsRussian)

{

MatrixRows = 4;

MatrixColumns = 8;

}

else

{

MatrixRows = 5;

MatrixColumns = 5;

}

(int rowA, int colA) = FindPosition(a);

(int rowB, int colB) = FindPosition(b);

if (rowA == rowB)

{

Debug.WriteLine(((colA - 1) % MatrixColumns + MatrixRows) % MatrixRows);

return $"{matrix[rowA, ((colA - 1) % MatrixColumns + MatrixColumns) % MatrixColumns]}{matrix[rowB, ((colB - 1) % MatrixColumns + MatrixColumns) % MatrixColumns]}";

}

else if (colA == colB)

{

return $"{matrix[((rowA - 1) % MatrixRows + MatrixRows) % MatrixRows, colA]}{matrix[((rowB - 1) % MatrixRows + MatrixRows) % MatrixRows, colB]}";

}

else

{

return $"{matrix[rowA, colB]}{matrix[rowB, colA]}";

}

}

private (int, int) FindPosition(char c)

{

int MatrixRows = 0;

int MatrixColumns = 0;

if (IsRussian)

{

MatrixRows = 4;

MatrixColumns = 8;

}

else

{

MatrixRows = 5;

MatrixColumns = 5;

}

for (int row = 0; row < MatrixRows; row++)

{

for (int col = 0; col < MatrixColumns; col++)

{

if (matrix[row, col] == c)

{

return (row, col);

}

}

}

throw new ArgumentException($"Символ {c} не найден в матрице");

}

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

{

key = PrepareKey(textBox2.Text);

GenerateMatrix();

string plaintext = PrepareText(textBox1.Text);

textBox4.Text = Decrypt(plaintext);

DisplayMatrixInListBox();

}

}

}