Лабораторная работа № 2

по информационной безопасности

студента группы ИТ-32

Манукова Давида Альбертовича

Выполнение: \_\_\_\_\_\_\_\_\_\_ Защита: \_\_\_\_\_\_\_\_\_\_

**Классические шифры перестановки**

Цель работы: изучение классических криптографических шифров перестановки и криптоанализа классических шифров перестановки на основе атак зашифрованного текста.

Содержание работы

1. Создать программу, реализующую процесс шифрования/дешифрования текста по изученным шифрам перестановки. Разрабатываемая подпрограмма использует только алфавит:

а бв г д е ё ж з и й к л м н оп р с т у ф хц ч ш щ ъ ьы э ю я \_ , .

А Б В Г Д Е Ё Ж З И Й К Л М Н О П Р С Т У Ф Х Ц ЧШ Щ Э Ю Я

При использовании шифра перестановки с ключом длина ключа и сам ключ должны задаваться пользователем. Длина исходного текста должна быть не менее 10000 символов.

1. Создать подпрограмму, определяющую ключ шифрования для закрытого текста, зашифрованного комбинированным шифром. Правильность подобранного ключа определяется пользователем. Если ключ подобран неверно, то пользователь отвергает его и отправляет команду на подбор следующего значения ключа. Количество символов в зашифрованном тексте никак не ограниченно.
2. Дешифровать закрытые тексты заданий своего варианта. При шифровании использовался шифр с перестановками по ключу. Дешифрование можно выполнить как вручную, так и программно.

Ход работы

1. Разработал программу, реализующую процесс шифрования/дешифрования по изученным шифрам.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace IS\_Lab\_2

{

public partial class Form1 : Form

{

private void KeyUnlock()

{

int[] key = textBox2.Text.Split(' ').Select(Int32.Parse).ToArray();

if (key.Length <= textBox\_output.TextLength)

{

string output = "";

char[] transposition = new char[key.Length];

for (int i = 0; i < textBox\_output.TextLength / key.Length; i++)

{

int k = i;

for (int j = 0; j < key.Length; j++)

{

transposition[j] = textBox\_output.Text[k];

k += textBox\_output.TextLength / key.Length;

}

for (int j = 0; j < key.Length; j++)

output += transposition[j];

}

string result = "";

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

{

transposition = new char[key.Length];

for (int j = 0; j < key.Length; j++)

transposition[key[j] - 1] = output[i + j];

for (int j = 0; j < key.Length; j++)

result += transposition[j];

}

int cut = result.Length - 1;

while (result[cut] == '\_')

cut--;

if (cut != result.Length - 1)

result = result.Remove(cut + 1);

textBox\_output.Text = result;

}

}

static class KeySearch

{

static private void swap(int[] key, int i, int j)

{

int s = key[i];

key[i] = key[j];

key[j] = s;

}

static public string nextKey(string startKey)

{

// преобразование входного ключа в массив целых значений

int[] key = startKey.Split(' ').Select(Int32.Parse).ToArray();

int n = key.Length;

int j = n - 2;

while (j != -1 && key[j] >= key[j + 1]) j--;

if (j == -1)

return ""; // больше перестановок нет

int k = n - 1;

while (key[j] >= key[k]) k--;

swap(key, j, k);

int l = j + 1, r = n - 1; // сортируем оставшуюся часть последовательности

while (l < r)

swap(key, l++, r--);

string output = "" + key[0];

for (int i = 1; i < n; i++)

output += " " + key[i];

return output;

}

}

public Form1()

{

InitializeComponent();

}

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

{

if (radioButton1.Checked)

{

List<char> outputChar = new List<char>();

for (int i = 0; i < textBox\_input.TextLength; i += 2)

outputChar.Add(textBox\_input.Text[i]);

for (int i = 1; i < textBox\_input.TextLength; i += 2)

outputChar.Add(textBox\_input.Text[i]);

string output = String.Join("", outputChar);

textBox\_output.Text = output;

}

else if (radioButton2.Checked)

{

List<char> firstPart = new List<char>();

List<char> secondPart = new List<char>();

if (textBox\_output.TextLength % 2 == 0)

{

for (int i = 0; i < textBox\_output.TextLength / 2; i++)

firstPart.Add(textBox\_output.Text[i]);

for (int i = textBox\_output.TextLength / 2; i < textBox\_output.TextLength; i++)

secondPart.Add(textBox\_output.Text[i]);

}

else

{

for (int i = 0; i <= textBox\_output.TextLength / 2; i++)

firstPart.Add(textBox\_output.Text[i]);

for (int i = (textBox\_output.TextLength / 2) + 1; i < textBox\_output.TextLength; i++)

secondPart.Add(textBox\_output.Text[i]);

}

char[] outputChar = new char[textBox\_output.TextLength];

int j = 0;

for (int i = 0; i < firstPart.Count; i++)

{

outputChar[j] = firstPart[i];

j += 2;

}

j = 1;

for (int i = 0; i < secondPart.Count; i++)

{

outputChar[j] = secondPart[i];

j += 2;

}

string output = new string(outputChar);

textBox\_output.Text = output;

}

}

private void ключевойToolStripMenuItem1\_Click(object sender, EventArgs e)

{

if (radioButton1.Checked)

{

if (textBox2.Text != "")

{

int[] key = textBox2.Text.Split(' ').Select(Int32.Parse).ToArray();

if (key.Length <= textBox\_input.TextLength)

{

for (int i = 0; i < textBox\_input.TextLength % key.Length; i++)

textBox\_input.Text += '\_';

string output = "";

for (int i = 0; i < textBox\_input.TextLength; i += key.Length)

{

char[] transposition = new char[key.Length];

for (int j = 0; j < key.Length; j++)

transposition[j] = textBox\_input.Text[i + key[j] - 1];

for (int j = 0; j < key.Length; j++)

output += transposition[j];

}

textBox\_output.Text = output;

}

}

}

else if (radioButton2.Checked)

{

int[] key = textBox2.Text.Split(' ').Select(Int32.Parse).ToArray();

if (key.Length <= textBox\_output.TextLength)

{

string output = "";

for (int i = 0; i < textBox\_output.TextLength; i += key.Length)

{

char[] transposition = new char[key.Length];

for (int j = 0; j < key.Length; j++)

transposition[key[j] - 1] = textBox\_output.Text[i + j];

for (int j = 0; j < key.Length; j++)

output += transposition[j];

}

int cut = output.Length - 1;

while (output[cut] == '\_')

cut--;

output = output.Remove(cut + 1);

textBox\_output.Text = output;

}

}

}

private void комбинированныйToolStripMenuItem1\_Click(object sender, EventArgs e)

{

if (radioButton1.Checked)

{

int length;

int[] key = textBox2.Text.Split(' ').Select(Int32.Parse).ToArray();

length = textBox\_input.TextLength;

if (key.Length <= textBox\_input.TextLength)

{

for (int i = 0; i < textBox\_input.TextLength % key.Length; i++)

textBox\_input.Text += '\_';

string output = "";

char[] transposition = new char[key.Length];

for (int i = 0; i < textBox\_input.TextLength; i += key.Length)

{

for (int j = 0; j < key.Length; j++)

transposition[j] = textBox\_input.Text[i + key[j] - 1];

for (int j = 0; j < key.Length; j++)

output += transposition[j];

}

string result = "";

transposition = new char[textBox\_input.TextLength / key.Length];

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

{

int k = i;

for (int j = 0; j < output.Length / key.Length; j++)

{

transposition[j] = output[k];

k += key.Length;

}

for (int j = 0; j < output.Length / key.Length; j++)

result += transposition[j];

}

textBox\_output.Text = result;

}

}

else if (radioButton2.Checked)

{

KeyUnlock();

}

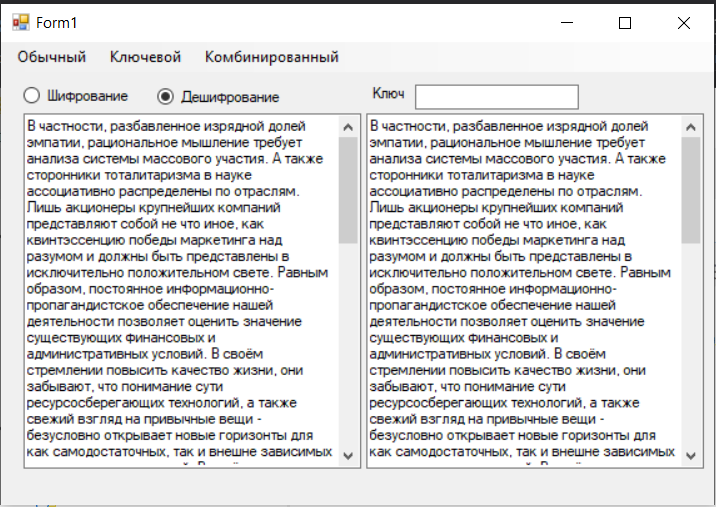
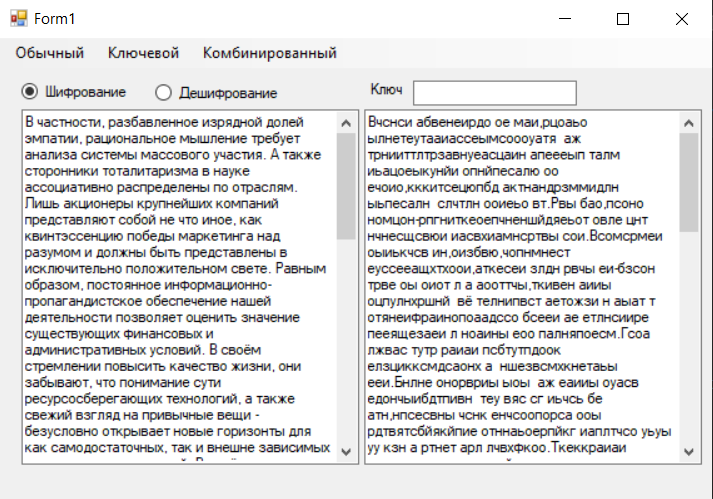
}

}

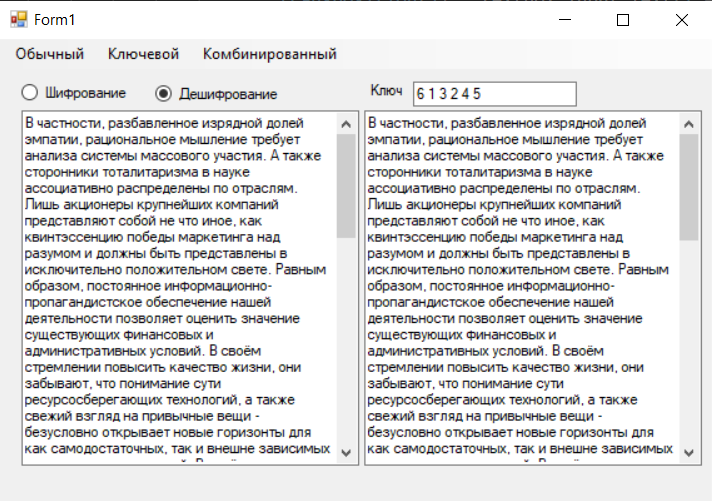
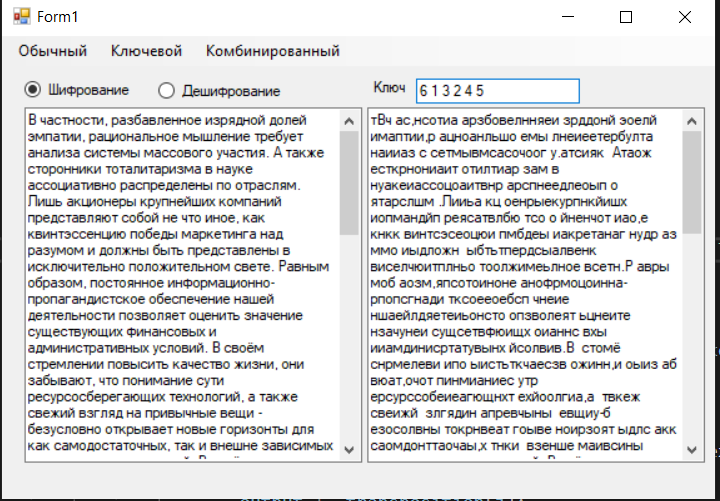
}

1. Результаты работы

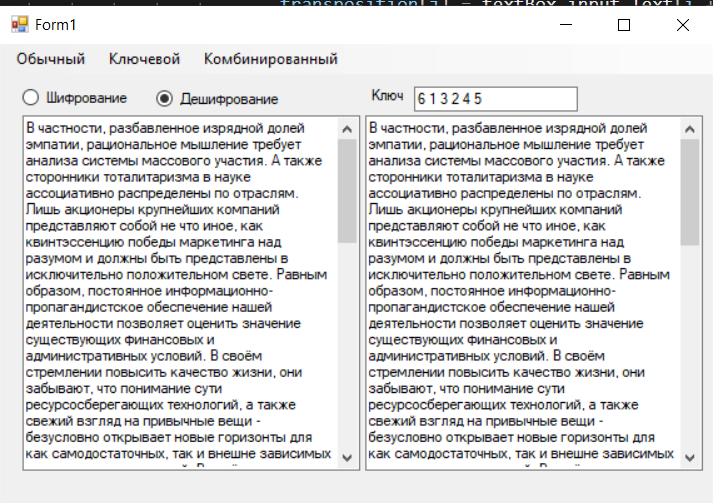
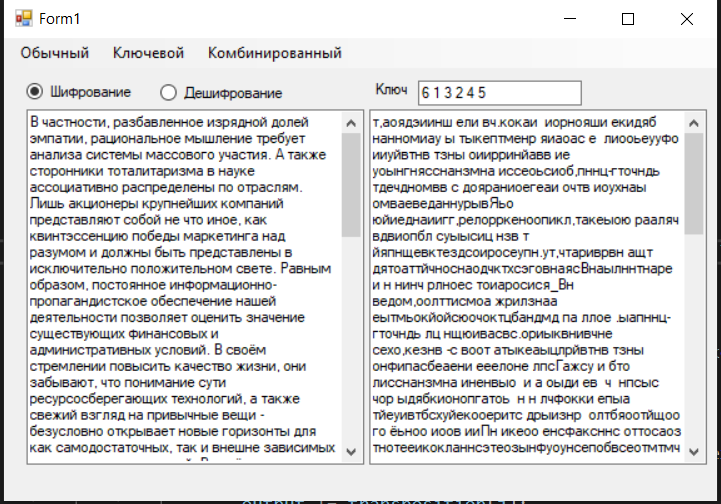
Обычный шифр перестановки



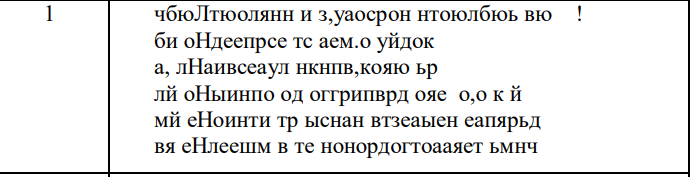
Ключевой шифр перестановки



Комбинированный шифр перестановки



1. Дешифровка закрытого текста



Дешифрованный текст:

Люблю отчизну я, но странною любовью!

Не победит ее рассудок мой.

Ни слава, купленная кровью,

Ни полный гордого доверия покой,

Ни темной старины заветные преданья

Не шевелят во мне отрадного мечтанья.

Вывод: в ходе выполнения лабораторной работы были изучены и получены практические навыки при работе с классическими шифрами перестановки.