

Федеральное государственное автономное  
образовательное учреждение  
высшего образования  
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт Космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №3**

Практическое задание №3 (основы статистического криптоанализа)

тема

Преподаватель

Студент КИ18-17/16 031831229

номер группы, зачетной книжки

подпись, дата

подпись, дата

А.А. Сидарас

инициалы, фамилия

В.А. Прекель

инициалы, фамилия

Красноярск 2020

## **1 Цель работы**

- введение в криптосистемы полиалфавитной подстановки на примере шифра Виженера;
- изучение методов криптоанализа шифров полиалфавитной подстановки на примере шифра Виженера.

## **2 Постановка задачи**

Используя любой из способов криптоанализа (рассмотренных ранее) шифров полиалфавитной подстановки расшифруйте текст сообщения (согласно вашему персональному варианту), а также определите секретный ключ, которым оно было зашифровано.

## **3 Краткий теоретический материал**

### **3.1 Подробное описание разработанной программы, включая краткое руководство пользователя (криптоаналитика) по её использованию**

В программе реализована возможность выбора шифротекста из любого варианта. После выбора варианта или ввода шифротекста, криптоаналитик должен нажать "Подобрать длину". После этого в выпадающем списке появятся возможные длины ключа, самая вероятная выберется сама. Далее криптоаналитик может нажать "Подобрать ключ", который подберётся, основываясь на то, что самая встречающаяся буква - "О". Далее при нажатии "Расшифровать" шифротекст расшифруется по подобранному ключу. Если текст осмысленный (а в заданных вариантах ещё и осмысленный ключ), то криптоанализ закончен. Иначе можно или подбирать осмысленный ключ, или подбирать другие часто встречающиеся буквы (в порядке убывания частоты - "ОЕАИНТСЛ...", или побуквенно изменять расшифрованный текст чтобы тот становился осмысленным, затем нажимая на "Исправить ключ".

### 3.2 Подробное и поэтапное описание процесса дешифрации, сделанного для взлома зашифрованного текста и получения секретного ключа

При нажатии на "Подобрать длину" происходит поиск длины ключа тестом Касиски. Для всех триграмм находится наибольший общий делитель расстояний между ними. Затем сортируются по частоте.

При нажатии на "Подобрать ключ" для каждого столбца шифротекста находится самая встречающаяся зашифрованная буква и основываясь на предположении, какая эта буква, вычисляется буква ключа.

При нажатии на "Исправить ключ" для первой несовпадающей буквы в изменённом тексте и тексте, расшифрованном прежним ключом, подбирается новая буква ключа.

## 4 Листинг составленной программы

### Листинг 1 – Lab\_03/Lab\_03.Core/VigenereDecrypter.cs

```
using System;
using System.Linq;
using System.Text;

namespace Lab_03.Core
{
    public class VigenereDecrypter
    {
        private const string RussianLetters =
"АБВГДЕЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";

        public VigenereDecrypter(string cipherTextOriginal)
        {
            CipherTextOriginal = cipherTextOriginal;
            CipherTextOnlyLetters = new
string(CipherTextOriginal.Where(Char.IsLetter).ToArray());
        }

        public string CipherTextOriginal { get; }

        public string CipherTextOnlyLetters { get; }

        public string TextOnlyLetters { get; private set; } = "";

        public string? Text
        {
            get
```

```

        {
            Decrypt();
            if (TextOnlyLetters.Length != CipherTextOnlyLetters.Length)
            {
                return null;
            }

            var sb = new StringBuilder();
            var k = 0;
            foreach (var i in CipherTextOriginal)
            {
                sb.Append(Char.IsLetter(i) ? TextOnlyLetters[k++] : i);
            }

            return sb.ToString();
        }
    }

    public string Key { get; set; } = "A";

    public static char Decrypt(char i, char p)
    {
        var e = RussianLetters.IndexOf(i);
        var k = RussianLetters.IndexOf(p);
        var m = RussianLetters.Length;

        var o = (e - k) % m;
        if (o < 0)
        {
            o = m + o;
        }

        var ch = RussianLetters[o];
        return ch;
    }

    public static char Crypt(char i, char p)
    {
        var e = RussianLetters.IndexOf(i);
        var k = RussianLetters.IndexOf(p);
        var m = RussianLetters.Length;

        var o = (e + k) % m;

        var ch = RussianLetters[o];
        return ch;
    }

    private void Decrypt()
    {
        var text = new StringBuilder();
        var j = 0;
        foreach (var i in CipherTextOnlyLetters)
        {
            var ch = Decrypt(i, Key[j++]);
            j %= Key.Length;
            var m = RussianLetters.Length;
            text.Append(ch);
        }

        TextOnlyLetters = text.ToString();
    }
}

```

```

    }
}

```

## Листинг 2 – Lab\_03/Lab\_03.Core/VigenereAnalysis.cs

```

using System;
using System.Collections.Generic;
using System.Diagnostics.CodeAnalysis;
using System.Linq;

namespace Lab_03.Core
{
    public class VigenereAnalysis
    {
        public VigenereAnalysis(string cipherText) => VigenereDecrypter = new
VigenereDecrypter(cipherText);

        public VigenereDecrypter VigenereDecrypter { get; }

        public int Mu { get; private set; }

        public string MostOccuringLettets => new string(Enumerable.Range(0, Mu)
            .Select(index => Enumerable.Range(0,
VigenereDecrypter.TextOnlyLetters.Length / Mu)
                .Select(i => VigenereDecrypter.TextOnlyLetters.Substring(i * Mu,
Mu))
                    .Select(i => i[index])
                    .Aggregate(new Dictionary<char, int>(),
                        (charcouner, c) =>
                        {
                            charcouner[c] = charcouner.ContainsKey(c) ?
charcouner[c] + 1 : 1;
                            return charcouner;
                        })
                    .OrderByDescending(pair => pair.Value)
                    .Select(pair => pair.Key)
                    .First())
                .ToArray());

        public void SuggestMu(int mu)
        {
            Mu = mu;
            VigenereDecrypter.Key = new string('A', mu);
        }

        public void SuggestMostOccuring(int index, char letter)
        {
            var most = Enumerable.Range(0,
VigenereDecrypter.CipherTextOnlyLetters.Length / Mu)
                .Select(i => VigenereDecrypter.CipherTextOnlyLetters.Substring(i
* Mu, Mu))
                    .Select(i => i[index])
                    .Aggregate(new Dictionary<char, int>(),
                        (charcouner, c) =>
                        {
                            charcouner[c] = charcouner.ContainsKey(c) ?
charcouner[c] + 1 : 1;
                            return charcouner;
                        })
                    .OrderByDescending(pair => pair.Value)
                    .First()

```

```

        .Key;

        var keychar = VigenereDecrypter.Decrypt(most, letter);

        VigenereDecrypter.Key =
            new string(VigenereDecrypter.Key.Select((c, i) => i == index ?
keychar : c).ToArray());
    }

    private static int GCD(IEnumerable<int> numbers) =>
numbers.Aggregate(GCD);

    private static int GCD(int a, int b) =>
        // ReSharper disable once TailRecursiveCall
        b == 0 ? a : GCD(b, a % b);

    [SuppressMessage("ReSharper", "PossibleMultipleEnumeration")]
    public static int MuFromIndeces(IEnumerable<int> indeces) =>
        GCD(indeces.Skip(1).Zip(indeces.SkipLast(1), (a, b) => a - b));

    public Dictionary<string, IEnumerable<int>>> Subgrams(int length)
    {
        var res = new Dictionary<string, IEnumerable<int>>>();
        for (var i = 0; i < VigenereDecrypter.CipherTextOnlyLetters.Length -
length; i++)
        {
            var tri = VigenereDecrypter.CipherTextOnlyLetters.Substring(i,
length);

            if (res.ContainsKey(tri))
            {
                continue;
            }

            var list = new List<int>();
            res[tri] = list;

            var k = 0;
            while (true)
            {
                var f = VigenereDecrypter.CipherTextOnlyLetters.IndexOf(tri,
k, StringComparison.Ordinal);
                if (f == -1)
                {
                    break;
                }

                list.Add(f);
                k = f + 1;
            }
        }

        return res;
    }

    public IEnumerable<int> PossibleMus() =>
        Enumerable.Range(5, 5)
            .GroupJoin(Subgrams(3)
                .Where(pair => pair.Value.Count() >= 2)
                .Select(pair => MuFromIndeces(pair.Value)), i => i, j => j,
(i, ints) => (i, ints.Count()))
            .OrderByDescending(i => i.Item2)
            .Select(i => i.i);

```

```

        public void FixKey(string post)
        {
            var pre = VigenereDecrypter.TextOnlyLetters;
            var postOnlyLetters = new
string(post.Where(Char.IsLetter).ToArray());
            var ind = -1;
            for (var i = 0; i < Math.Min(pre.Length, postOnlyLetters.Length);
i++)
            {
                if (pre[i] == postOnlyLetters[i])
                {
                    continue;
                }

                ind = i;
                break;
            }

            var k = ind % Mu;

            var keychar = VigenereDecrypter.Decrypt(VigenereDecrypter.Key[k],
VigenereDecrypter.Decrypt(postOnlyLetters[ind], pre[ind]));

            VigenereDecrypter.Key =
                new string(VigenereDecrypter.Key.Select((c, i) => i == k ?
keychar : c).ToArray());
        }
    }
}

```

### Листинг 3 – Lab\_03/Lab\_03.Core/Variants.cs

```

namespace Lab_03.Core
{
    public static class Variants
    {
        private static readonly string[] data =
        {
            // Варианты...
        };

        private static readonly string[] answers =
        {
            "КОЛОДЕЦ",
            "ЗАНОЗА",
            "СЦЕНА",
            "ЛАЗУРЬ",
            "РЕЧНОЙ",
            "ЛАПОТЬ",
            "ЧЕЛОВЕК",
            "АРМИЯ",
            "ДИПЛОМ",
            "САМЫЙ",
            "ТРУДНЫЙ",
            "ЧЕСТЬ",
            "КЛОУНЫ",
            "КРУЖКА",
            "ЗАМЫСЕЛ"
        };

        public static string GetVariantByNumber(int number) => data[number - 1];
    }
}

```

```

        public static int GetVariantsCount() => data.Length;

        public static string GetAnswerByNumber(int number) => answers[number -
1];
    }
}

```

#### Листинг 4 – Lab\_03/Lab\_03.App/MainWindow.xaml.cs

```

using System;
using System.Linq;

using Avalonia.Controls;
using Avalonia.Interactivity;
using Avalonia.Markup.Xaml;

using Lab_03.Core;

using MessageBox.Avalonia;

namespace Lab_03.App
{
    public class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();

            private VigenerereAnalysis VigenerereAnalysis { get; set; } = new
VigenerereAnalysis("");

            private TextBox TextTextBox => this.FindControl<TextBox>("TextTextBox");
            private TextBox CiphertextTextBox =>
this.FindControl<TextBox>("CiphertextTextBox");
            private ComboBox VariantsComboBox =>
this.FindControl<ComboBox>("VariantsComboBox");
            private ComboBox MuComboBox => this.FindControl<ComboBox>("MuComboBox");
            private TextBox MostOcurrredLettersTextBox =>
this.FindControl<TextBox>("MostOcurrredLettersTextBox");
            private TextBox KeyTextBox => this.FindControl<TextBox>("KeyTextBox");

            private void InitializeComponent()
            {
                AvaloniaXamlLoader.Load(this);

                VariantsComboBox.Items = Enumerable.Range(1,
Variants.GetVariantsCount());
            }

            private void OnVariantsComboBoxSelectionChanged(object sender,
SelectionChangedEventArgs e)
            {
                try
                {
                    CiphertextTextBox.Text = Variants.GetVariantByNumber((int)
VariantsComboBox.SelectedItem);
                }
                catch (Exception ex)

```



```

        {
            ExceptionMessageBox(ex);
        }
    }

    private void OnMuComboBoxSelectionChanged(object sender,
        SelectionChangedEventArgs e)
    {
        try
        {
            if (MuComboBox.SelectedItem == null)
            {
                return;
            }

            VigenereAnalysis.SuggestMu((int) MuComboBox.SelectedItem);
            KeyTextBox.Text = VigenereAnalysis.VigenereDecrypter.Key;
            MostOcurrredLettersTextBox.Text = new string('O',
VigenereAnalysis.Mu);
        }
        catch (Exception ex)
        {
            ExceptionMessageBox(ex);
        }
    }

    private void OnPossibleMusButtonClick(object sender, RoutedEventArgs e)
    {
        try
        {
            VigenereAnalysis = new VigenereAnalysis(CiphertextTextBox.Text);
            var items = VigenereAnalysis.PossibleMus().ToList();
            MuComboBox.Items = items;
            MuComboBox.SelectedItem = items.First();

            VigenereAnalysis.SuggestMu(items.First());
            KeyTextBox.Text = VigenereAnalysis.VigenereDecrypter.Key;
            MostOcurrredLettersTextBox.Text = new string('O',
VigenereAnalysis.Mu);
        }
        catch (Exception ex)
        {
            ExceptionMessageBox(ex);
        }
    }

    private void OnSuggestMostOccuredButtonClick(object sender,
        RoutedEventArgs e)
    {
        try
        {
            for (var i = 0; i < VigenereAnalysis.Mu; i++)
            {
                VigenereAnalysis.SuggestMostOccuring(i,
MostOcurrredLettersTextBox.Text[i]);
            }

            KeyTextBox.Text = VigenereAnalysis.VigenereDecrypter.Key;
        }
        catch (Exception ex)
        {
            ExceptionMessageBox(ex);
        }
    }

```

```

    }

    private void OnEncryptButtonClick(object sender, RoutedEventArgs e)
    {
        try
        {
            VigenereAnalysis.VigenereDecrypter.Key = KeyTextBox.Text;
            TextTextBox.Text = VigenereAnalysis.VigenereDecrypter.Text;
            MostOcurrredLettersTextBox.Text =
VigenereAnalysis.MostOccuringLettets;
        }
        catch (Exception ex)
        {
            ExceptionMessageBox(ex);
        }
    }

    private void OnFixKeyButtonClick(object sender, RoutedEventArgs e)
    {
        try
        {
            VigenereAnalysis.FixKey(TextTextBox.Text);
            KeyTextBox.Text = VigenereAnalysis.VigenereDecrypter.Key;
            TextTextBox.Text = VigenereAnalysis.VigenereDecrypter.Text;
            MostOcurrredLettersTextBox.Text =
VigenereAnalysis.MostOccuringLettets;
        }
        catch (Exception ex)
        {
            ExceptionMessageBox(ex);
        }
    }

    private void OnCheckKeyButtonClick(object sender, RoutedEventArgs e)
    {
        try
        {
            VigenereAnalysis.VigenereDecrypter.Key =
                Variants.GetAnswerByNumber((int)
VariantsComboBox.SelectedItem);
            KeyTextBox.Text = VigenereAnalysis.VigenereDecrypter.Key;
            TextTextBox.Text = VigenereAnalysis.VigenereDecrypter.Text;
            MostOcurrredLettersTextBox.Text =
VigenereAnalysis.MostOccuringLettets;
        }
        catch (Exception ex)
        {
            ExceptionMessageBox(ex);
        }
    }

    private static void ExceptionMessageBox(Exception ex)
    {
        var msgBox = MessageBoxManager
            .GetMessageBoxStandardWindow("Ошибка", ex.Message + "\n" +
ex.StackTrace);
        msgBox.Show();
    }
}

```

## Листинг 5 – Lab\_03/Lab\_03.Core.Tests/VigenereTests.cs

```
using System.Linq;

using NUnit.Framework;

namespace Lab_03.Core.Tests
{
    public class VigenereTests
    {
        [Test]
        public void TestSample1()
        {
            var c = new VigenereDecrypter("ВЙИИОЗР") {Key = "АВВ"};

            Assert.That(c.Text, Is.EqualTo("ВИЖИИЕР"));
        }

        [Test]
        public void TestVar1()
        {
            var t = Variants.GetVariantByNumber(1);
            var an = new VigenereAnalysis(t);

            var pm = an.PossibleMus();

            an.SuggestMu(pm.First());
            an.SuggestMostOccuring(0, 'O');
            an.SuggestMostOccuring(1, 'O');
            an.SuggestMostOccuring(2, 'E');
            an.SuggestMostOccuring(3, 'O');
            an.SuggestMostOccuring(4, 'O');
            an.SuggestMostOccuring(5, 'O');
            an.SuggestMostOccuring(6, 'O');

            Assert.That(an.VigenereDecrypter.Key, Is.EqualTo("КОЛОДЕЦ"));

            Assert.That(an.VigenereDecrypter.Text?.Split(" ").First(),
                Is.EqualTo("МОЛОДОЙ"));
        }

        [Test]
        public void TestVar14()
        {
            var v14 = Variants.GetVariantByNumber(14);
            var v = new VigenereDecrypter(v14);

            var an = new VigenereAnalysis(v14);
            var pm = an.PossibleMus();

            an.SuggestMu(pm.First());
            an.SuggestMostOccuring(0, 'O');
            an.SuggestMostOccuring(1, 'O');
            an.SuggestMostOccuring(2, 'O');
            an.SuggestMostOccuring(3, 'O');
            an.SuggestMostOccuring(4, 'O');
            an.SuggestMostOccuring(5, 'O');

            Assert.That(an.VigenereDecrypter.Key, Is.EqualTo("КРУЖКА"));

            Assert.That(an.VigenereDecrypter.Text?.Split(" ").Skip(1).First(),
                Is.EqualTo("ПОЧИТАЮ"));
        }
    }
}
```

```

    }

[Test]
public void TestVar12()
{
    var v12 = Variants.GetVariantByNumber(12);
    var an = new VigenereAnalysis(v12);
    var pm = an.PossibleMus();

    an.SuggestMu(pm.First());
    an.SuggestMostOccuring(0, 'E');
    an.SuggestMostOccuring(1, 'A');
    an.SuggestMostOccuring(2, 'O');
    an.SuggestMostOccuring(3, 'O');
    an.SuggestMostOccuring(4, 'O');

    Assert.That(an.VigenereDecrypter.Key, Is.EqualTo("ЧЕСТЦ"));
    Assert.That(an.VigenereDecrypter.Text?.Split(" ").Skip(1).First(),
Is.EqualTo("ДВЕУАДЦАШЬ"));

    const string post = "В ДВЕНАДЦАШЬ";
    an.FixKey(post);

    Assert.That(an.VigenereDecrypter.Key, Is.EqualTo("ЧЕСТЬ"));
    Assert.That(an.VigenereDecrypter.Text?.Split(" ").Skip(1).First(),
Is.EqualTo("ДВЕНАДЦАТЬ"));
}

[Test]
public void TestAnswers()
{
    var starts = new[]
    {
        "МОЛОДОЙ ЧАРТКОВ",
        "ОДИНОКО В СТОРОНЕ",
        "ДЕД МОЙ",
        "КАК ТОЛЬКО РЫЦАРЬ",
        "НЕБО ЗВЕЗДИЛОСЬ",
        "ОТРЯД МИНУЛ ГОРОД",
        "Я ОЧЕНЬ ЛЮБЛЮ",
        "НИГДЕ НЕ ОСТАНАВЛИВАЛОСЬ",
        "ЗА САДОМ НАХОДИЛСЯ",
        "САМОЕ ТОРЖЕСТВЕННОЕ",
        "ФИЛОСОФ НАЧАЛ НА",
        "В ДВЕНАДЦАТЬ ЧАСОВ",
        "НО ПРЕЖДЕ НЕЖЕЛИ",
        "Я ПОЧИТАЮ НЕ ИЗЛИШНИМ",
        "С ДОСАДОЮ ЗАКУСИВ"
    };
    for (var i = 0; i < Variants.GetVariantsCount(); i++)
    {
        var v = Variants.GetVariantByNumber(i + 1);
        var an = new VigenereAnalysis(v);

        an.VigenereDecrypter.Key = Variants.GetAnswerByNumber(i + 1);

        Assert.That(an.VigenereDecrypter.Text?.Substring(0,
starts[i].Length), Is.EqualTo(starts[i]));
    }
}
}
}

```

## 5 Результат работы программы

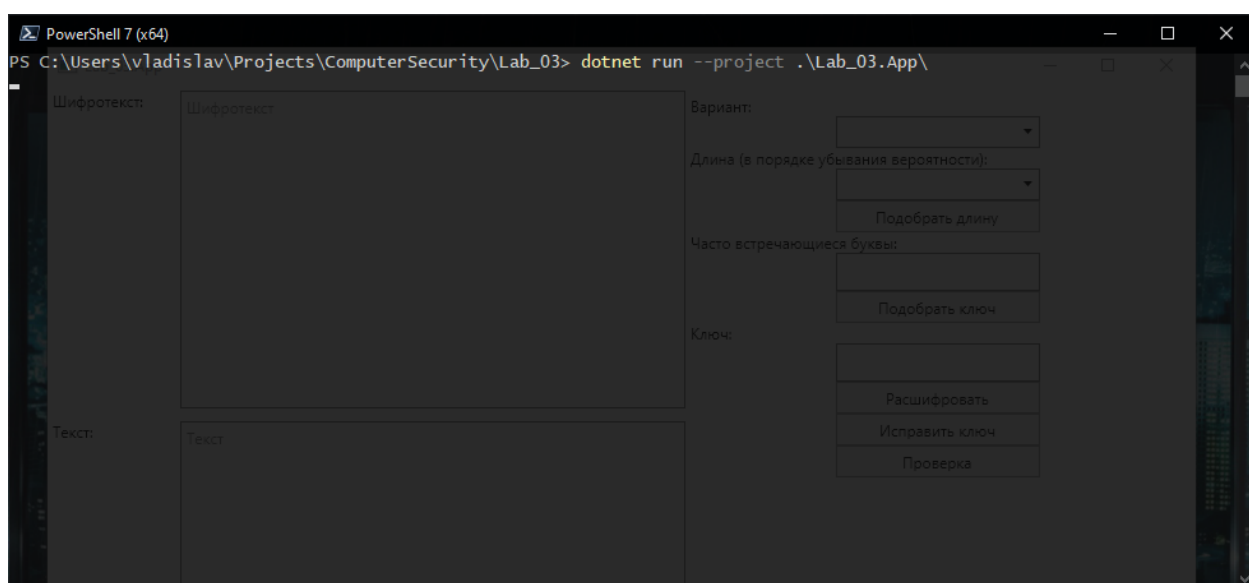


Рисунок 1 – Запуск приложения

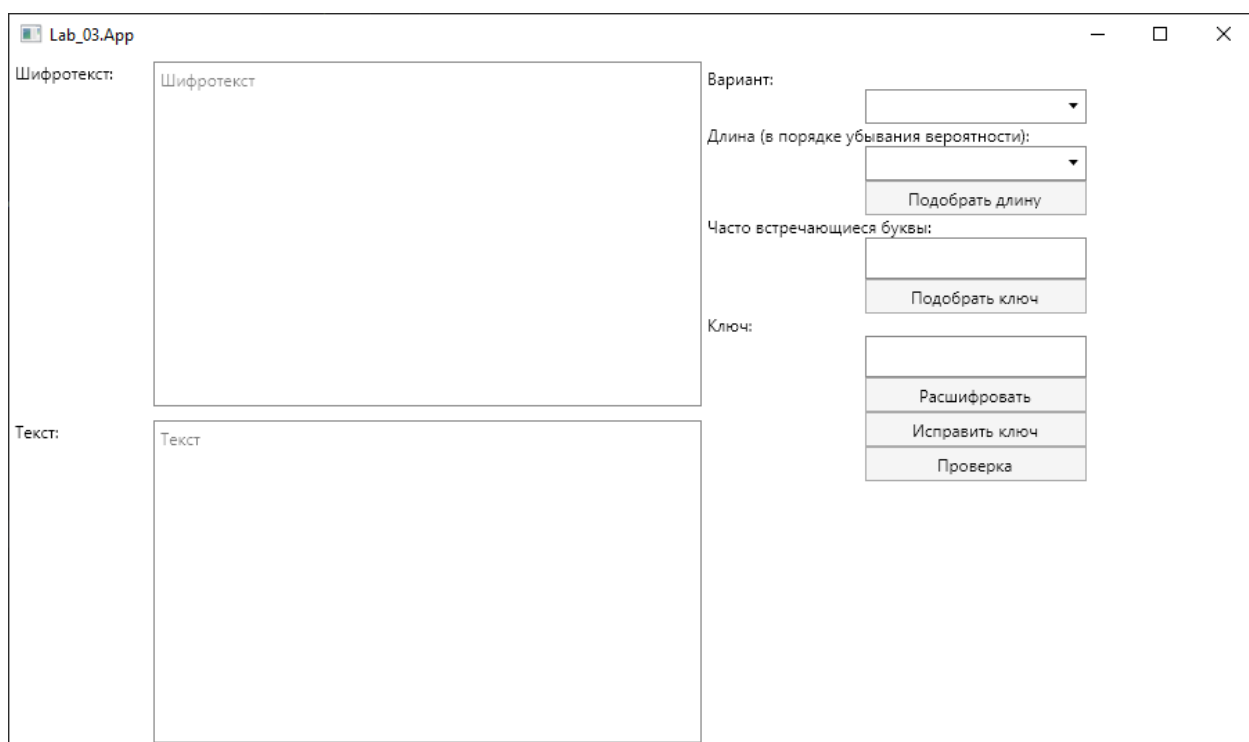


Рисунок 2 – Начальное окно

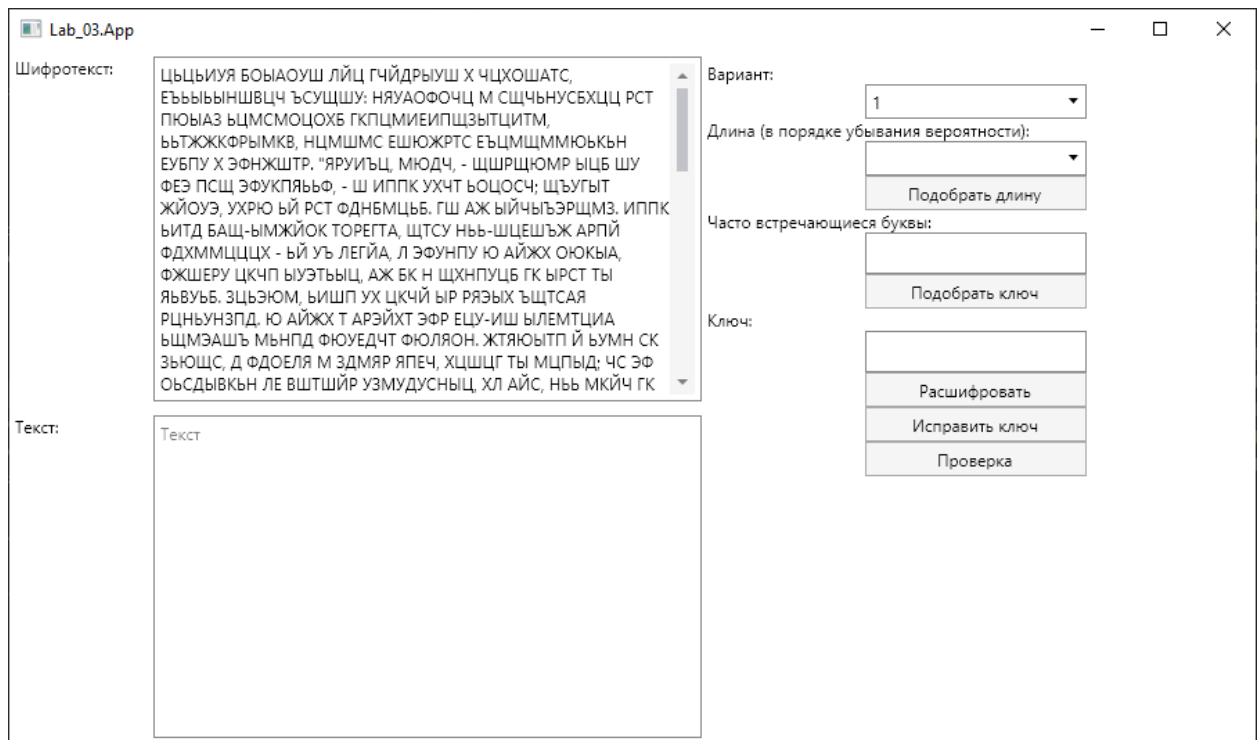


Рисунок 3 – После выбора варианта

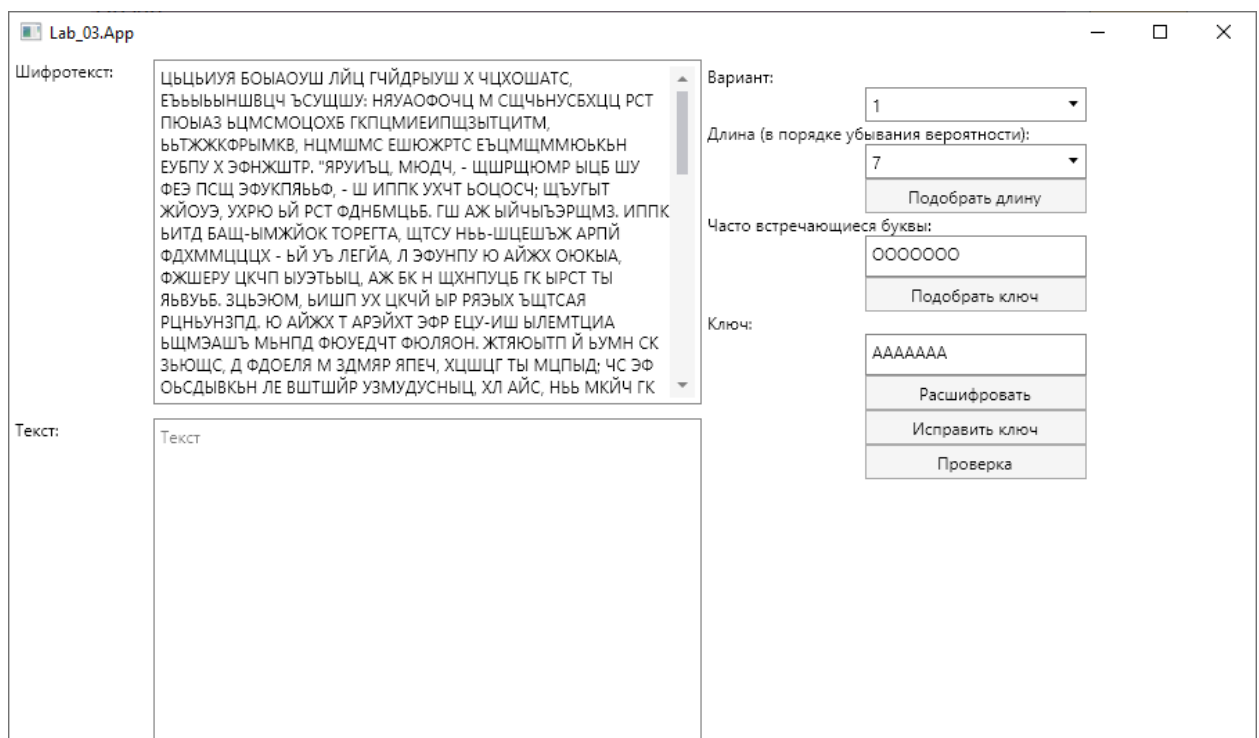


Рисунок 4 – После "Подобрать длину"

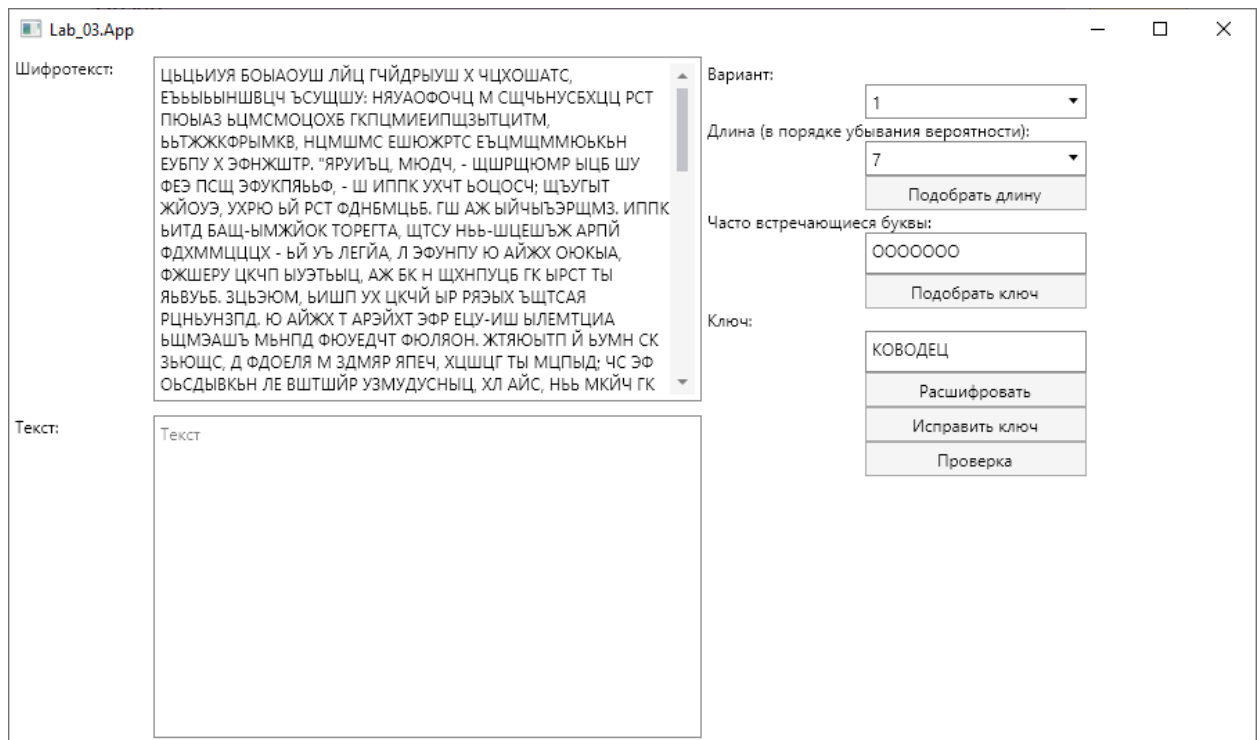


Рисунок 5 – После "Подобрать ключ"

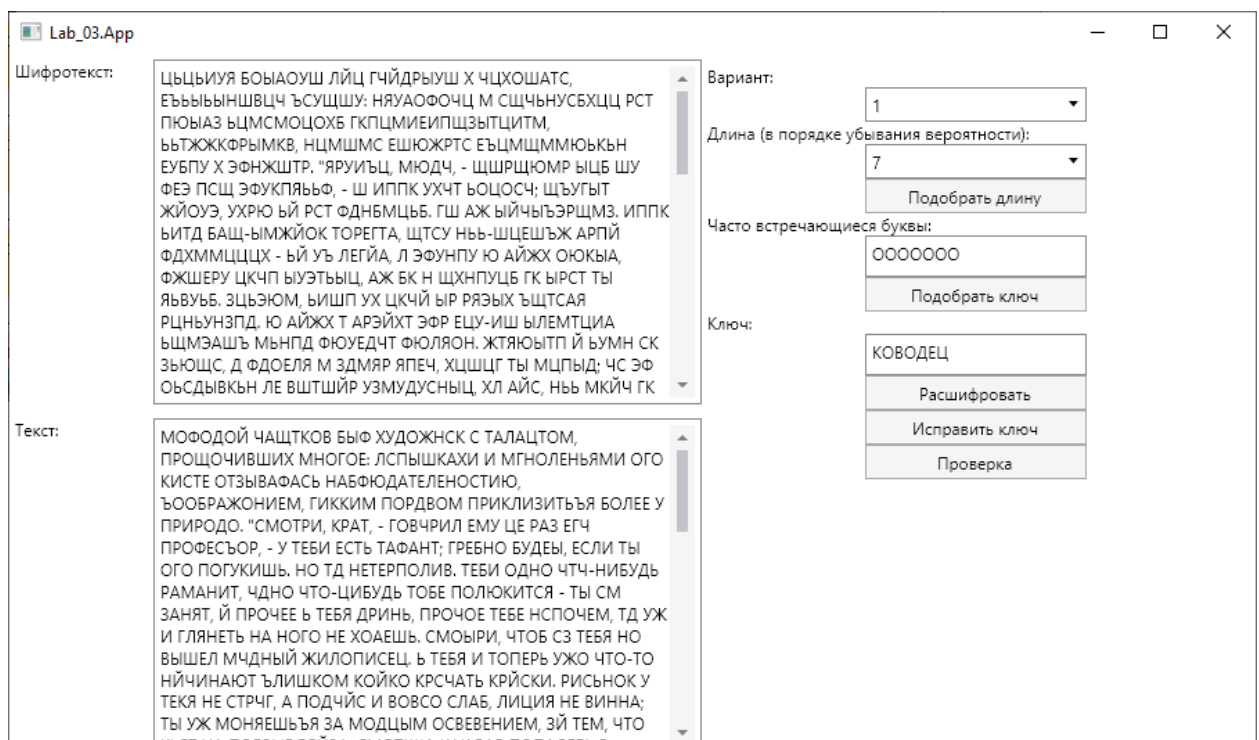


Рисунок 6 – После "Расшифровать"

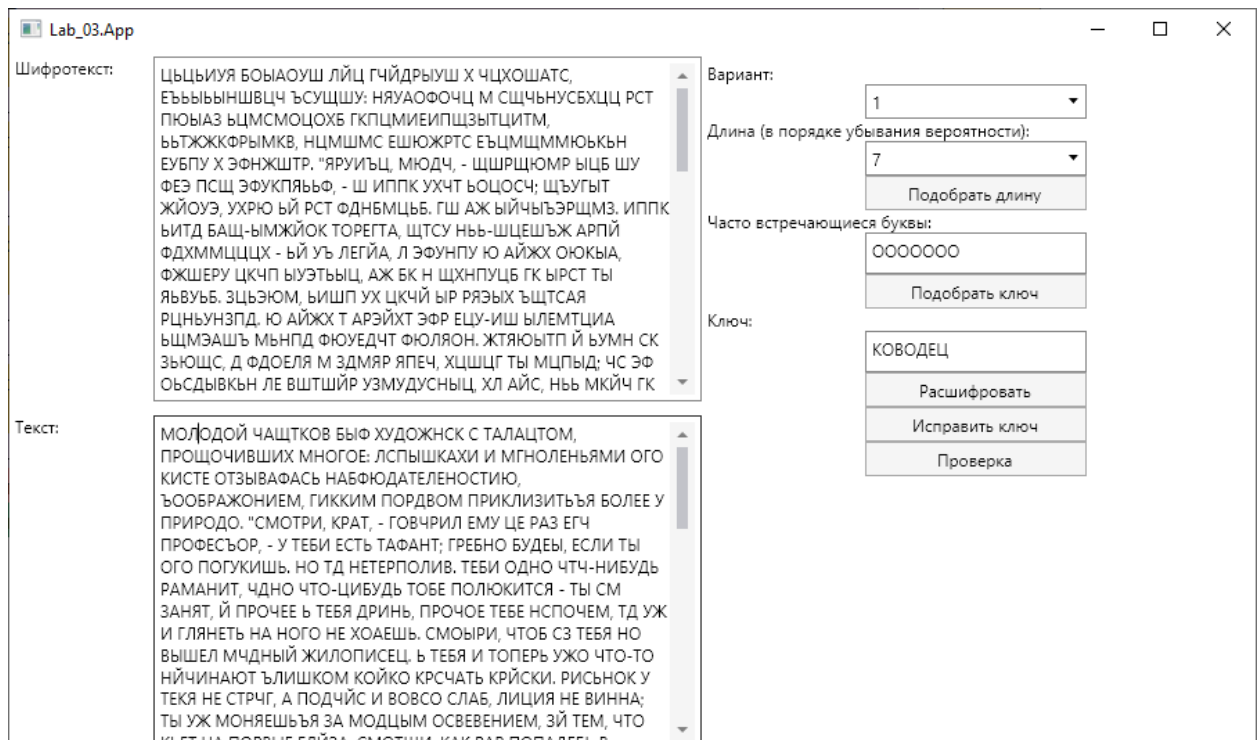


Рисунок 7 – После замены "МОФОДОЙ" на "МОЛОДОЙ"

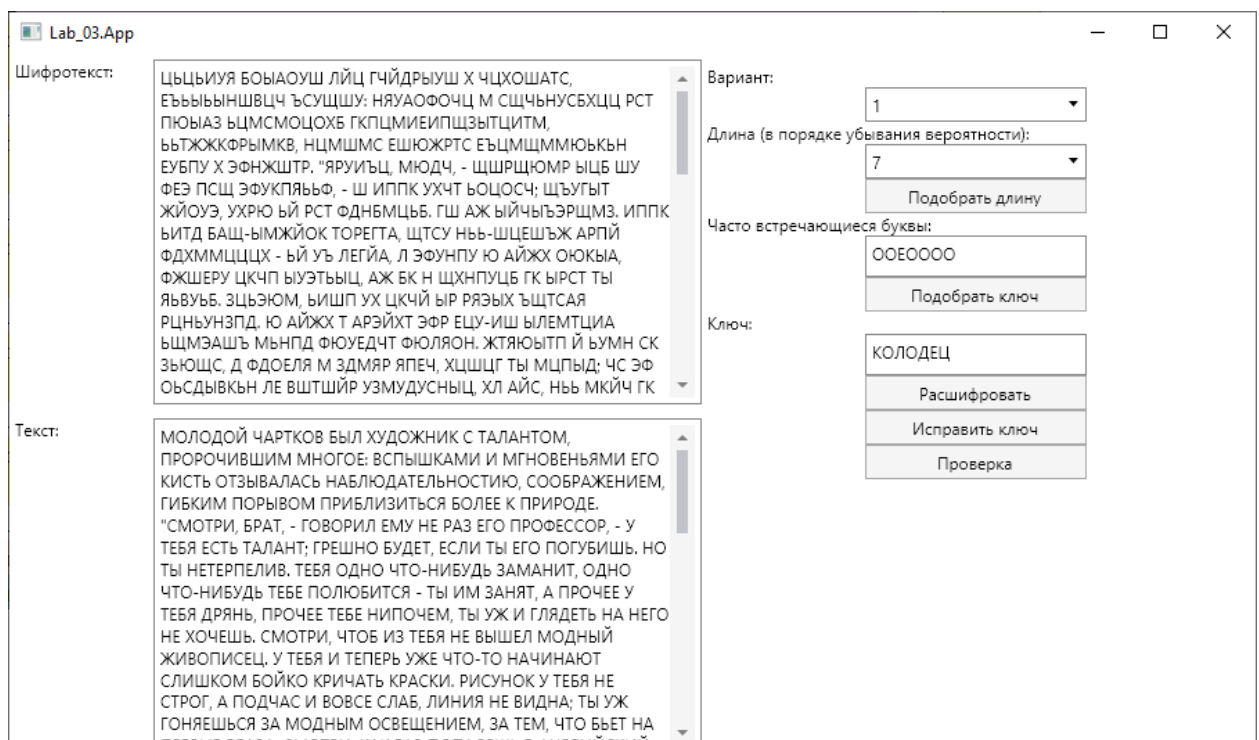
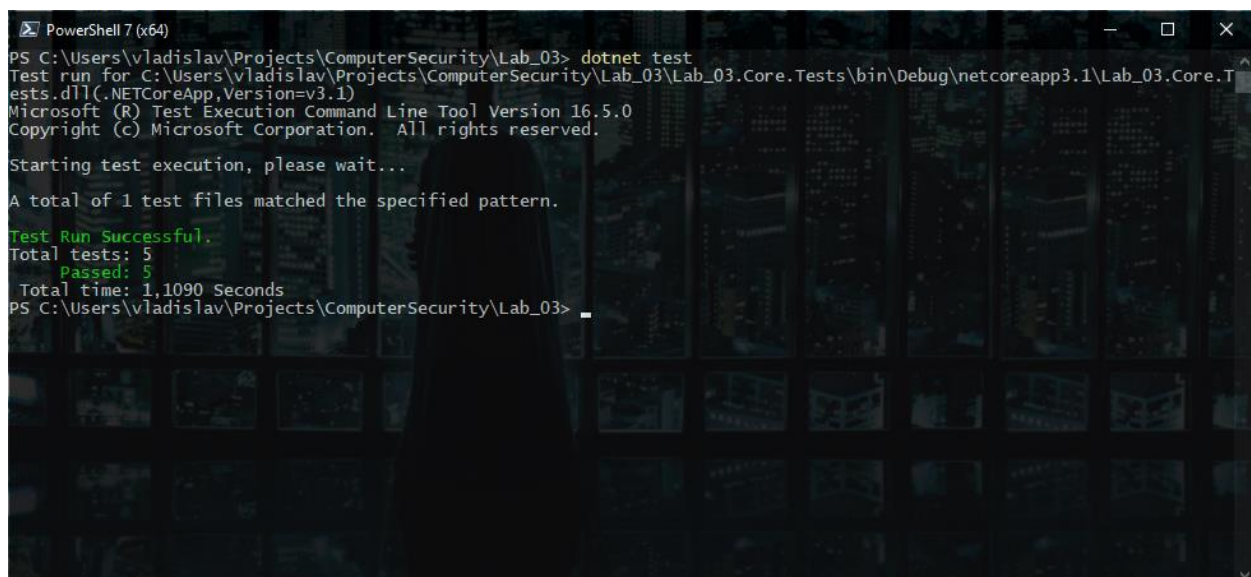


Рисунок 8 – После "Исправить ключ"





```
PowerShell 7 (x64)
PS C:\Users\vladislav\Projects\ComputerSecurity\Lab_03> dotnet test
Test run for C:\Users\vladislav\Projects\ComputerSecurity\Lab_03\Lab_03.Core.Tests\bin\Debug\netcoreapp3.1\Lab_03.Core.Tests.dll(.NETCoreApp,Version=v3.1)
Microsoft (R) Test Execution Command Line Tool Version 16.5.0
Copyright (c) Microsoft Corporation. All rights reserved.

Starting test execution, please wait...

A total of 1 test files matched the specified pattern.

Test Run Successful.
Total tests: 5
     Passed: 5
Total time: 1.1090 Seconds
PS C:\Users\vladislav\Projects\ComputerSecurity\Lab_03>
```

Рисунок 9 – Запуск тестов

## 6 Заключение

Были изучены основы криптоанализа на примере криптоанализа шифра Вижинера. Разработана программа, помогающая расшифровать текст на русском языке, зашифрованным шифром Вижинера. Написаны юнит-тесты.