Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Пермский национальный исследовательский политехнический университет»

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы»

Направление 09.03.01 – «Информатика и вычислительная техника»

Дисциплина: «Защита информации»

Профиль: «Автоматизированные системы обработки информации и управления»

Семестр 6

ОТЧЕТ

по лабораторной работе №8

Тема: «Методы гаммирования»

Вариант 4

Выполнил: студент группы АСУ-19-1б

Шеретов М.А. \_\_\_\_\_\_\_\_\_\_\_

Проверил: доцент кафедры ИТАС

Шереметьев В. Г. \_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_

Пермь, 2022

**ЦЕЛЬ РАБОТЫ**

Получить практические навыки по применению метода шифрования однократной случайной равновероятной гаммой – однократного гаммирования

**ЗАДАНИЕ**

Реализовать шифрование сообщения методом однократного гаммирования, используя блоки открытого текста длиной 48 бит и используя в алгоритме шифрования операцию сложения.

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**Шифрование методом гаммирования**

Под гаммированием понимают процесс наложения по определенному закону гаммы шифра на открытые данные. Гамма шифра — это псевдослучайная последовательность, выработанная по заданному алгоритму для зашифрования открытых данных и расшифрования зашифрованных данных.

Процесс зашифрования заключается в генерации гаммы шифра и наложении полученной гаммы на исходный открытый текст обратимым образом, например с использованием операции сложения по модулю 2.

Следует отметить, что перед зашифрованием открытые данные разбивают на блоки https://lh3.googleusercontent.com/4Mgn0KAkTRUimtUCNXrmgnzYjOF4heD5web8beWwGFWHT4dhYIbKQRcaCD5t7V301IaUUCX4r0uLu4_r43hagCAtYJLYZA3U3AOMqbA6pHjkaUS6FiYoKYwsd8jHlGS1pKxwQ2g одинаковой длины, обычно по 64 бита. Гамма шифра вырабатывается в виде последовательности блоковhttps://lh3.googleusercontent.com/HftXYJ1k9MTiG5CTQ2eWk4rlvOq1rz8S6U97wQtlS4lZucszRDJfBRdDS1a80FTd-hnsAn_G4fxfhV8IipEYRaJYr7iPr480VeRZTjC46tNvz5v8m6meyo3XpXCn_zKNjI8o-qQ аналогичной длины.

Уравнение зашифрования можно записать в виде

https://lh3.googleusercontent.com/-_hriVP3cuZ64v38QWO_tMMgtfa8MCt0BD-Nq_zdcxbtW3Vr_n43xS-8r2r6ADM5B2N2SqzX9nlUwuky1FWEzP5dULnI3nj1IX2tjmfOZBDAmXH4PmskqttC9JqO-uNM58AGgSs

где https://lh6.googleusercontent.com/cKg36L_qTsd63grRYnJfBHMA-4Mr2SpA1leZmvSf_WuZiku5g2lopfKgqCgywBlbYr_0CIbnyBMJqen515VaTWd9IHcpbQ9F-0Qes2jScmEmwDcysTjtdfYv8b5Da8lQxzKKR6g - *i*-й блок шифртекста; https://lh5.googleusercontent.com/fmCD6zdnK9VyclYUEAYI6zy1zd2NJQGz3vXf_Epz49qNabcwVHoss6rzA6Xz5hRh8y7vrrVelzhjSJsi7RoCJ8gS1H1lxvC-2bkIWl5jd7GpBj92o16CEcL4IC5kXClVKWhR45g- *i*-й блок гаммы шифра; https://lh5.googleusercontent.com/nBs2ZgxSB70J80mRpjV-U3a1khQfp56an7BD43KSpF93ZvHywqRcP4z2e0ozAo8Qnj1qMpYLHFcLW0hM22wCmNQnTpFJ2WVvvW7QQ23IhvQqrcebiz9XZLRK0A29iIyJKANe_II- *i*-й блок открытого текста; *М* - количество блоков открытого текста.

Процесс расшифрования сводится к повторной генерации гаммы шифра и наложению этой гаммы на зашифрованные данные. Уравнение расшифрования имеет вид обратный вид.

Получаемый этим методом шифртекст достаточно труден для раскрытия, поскольку теперь ключ является переменным. По сути дела гамма шифра должна изменяться случайным образом для каждого шифруемого блока. Если период гаммы превышает длину всего шифруемого текста и злоумышленнику неизвестна никакая часть исходного текста, то такой шифр можно раскрыть только прямым перебором всех вариантов ключа. В этом случае криптостойкость шифра определяется длиной ключа.

**«Разоблачение» гаммирования.**

С точки зрения теории криптоанализа метод шифрования однократной случайной равновероятной гаммой той же длины, что и открытый текст, является невскрываемым (далее для краткости авторы будут употреблять термин «однократное гаммирование», держа в уме все вышесказанное). Обоснование, которое привел Шеннон, основываясь на введенном им же понятии информации, не дает возможности усомниться в этом - из-за равных априорных вероятностей криптоаналитик не может сказать о дешифровке, верна она или нет. Кроме того, даже раскрыв часть сообщения, дешифровщик не сможет хоть сколько-нибудь поправить положение - информация о вскрытом участке гаммы не дает информации об остальных ее частях.

Логично было бы предположить, что для организации канала конфиденциальной связи в открытых сетях следовало бы воспользоваться именно схемой шифрования однократного гаммирования. Ее преимущества вроде бы очевидны. Есть, правда, один весомый недостаток, который сразу бросается в глаза, - это необходимость иметь огромные объемы данных, которые можно было бы использовать в качестве гаммы. Для этих целей обычно пользуются датчиками настоящих случайных чисел (в западной литературе аналогичный термин носит название TrueRandomNumberGenerator или TRNG). Это уже аппаратные устройства, которые по запросу выдают набор случайных чисел, генерируя их с помощью очень большого количества физических параметров окружающей среды. Статистические характеристики таких наборов весьма близки к характеристикам "белого шума", что означает равновероятное появление каждого следующего числа в наборе. А это, в свою очередь, означает для нас действительно равновероятную гамму.

К сожалению, для того чтобы организовать конфиденциальный канал передачи данных, потребуется записать довольно большое количество этих данных и обменяться ими по секретному каналу. Уже одно это условие делает однократное гаммирование во многих случаях неприемлемым. В самом деле, зачем передавать что-то по открытому незащищенному каналу, когда есть возможность передать все это по секретному защищенному? И хотя на простой вопрос, является ли метод использования однократной случайной равновероятной гаммы стойким к взлому, существует положительный ответ, его использование может оказаться попросту невозможным.

Да и к тому же метод однократного гаммированиякриптостоек только в определенных, можно даже сказать, тепличных условиях. Что же касается общего случая, то все не так просто.

**ХОД РАБОТЫ**

Форма приложения изображена на рисунке 1.

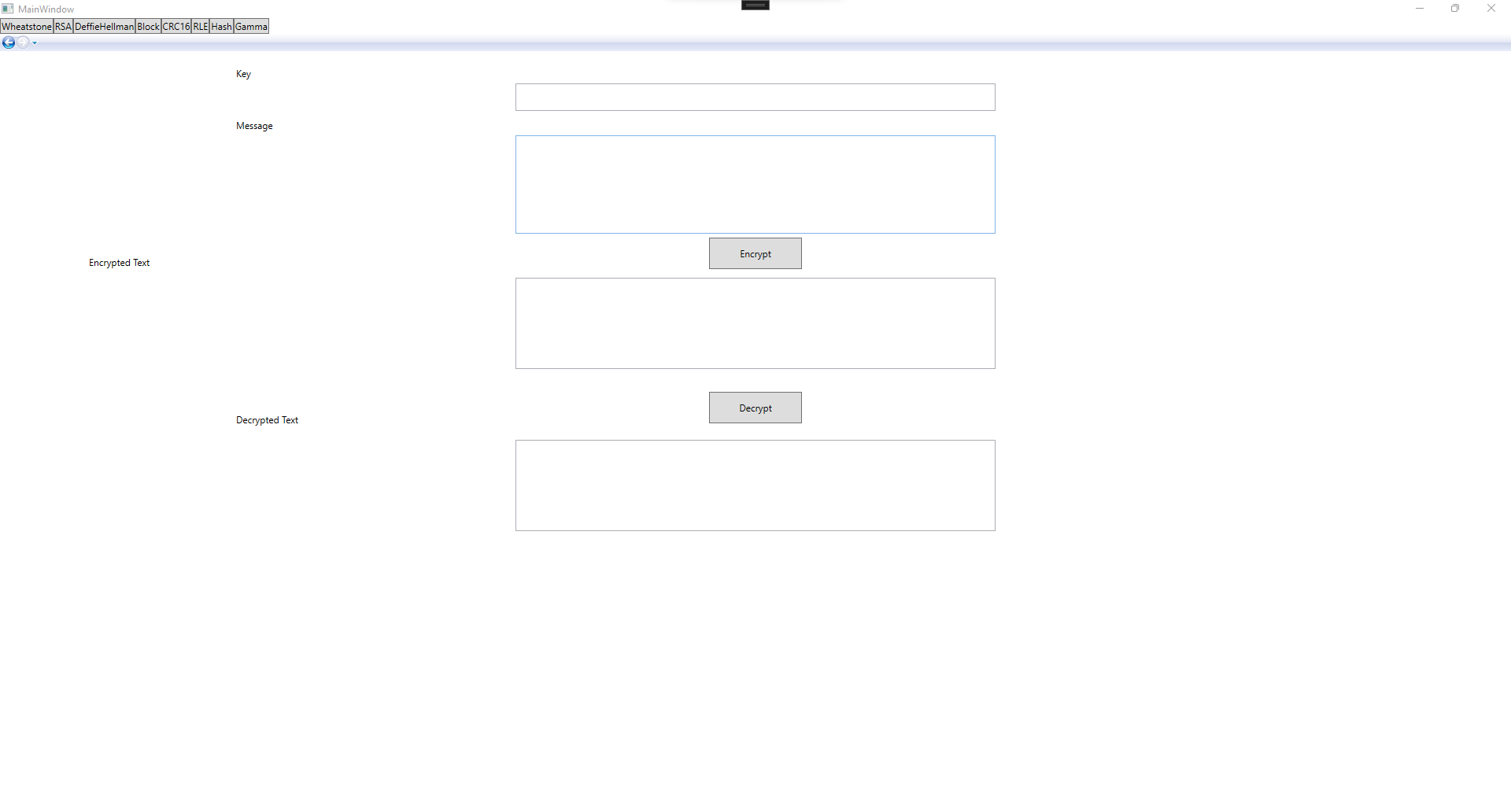


Рисунок 1 – Главное окно программы

В поле *Ключ* вводится ключ (гамма). Во поле *Исходный текст* вводится шифруемое сообщение. При нажатии на кнопку *Зашифровать* сообщение будет зашифровано с помощью операции XOR и записано в поле *Зашифрованный текст*. При том же ключе зашифрованную запись можно расшифровать нажатием кнопки *Расшифровать*.

Примеры работы программы представлен на рисунке 2.

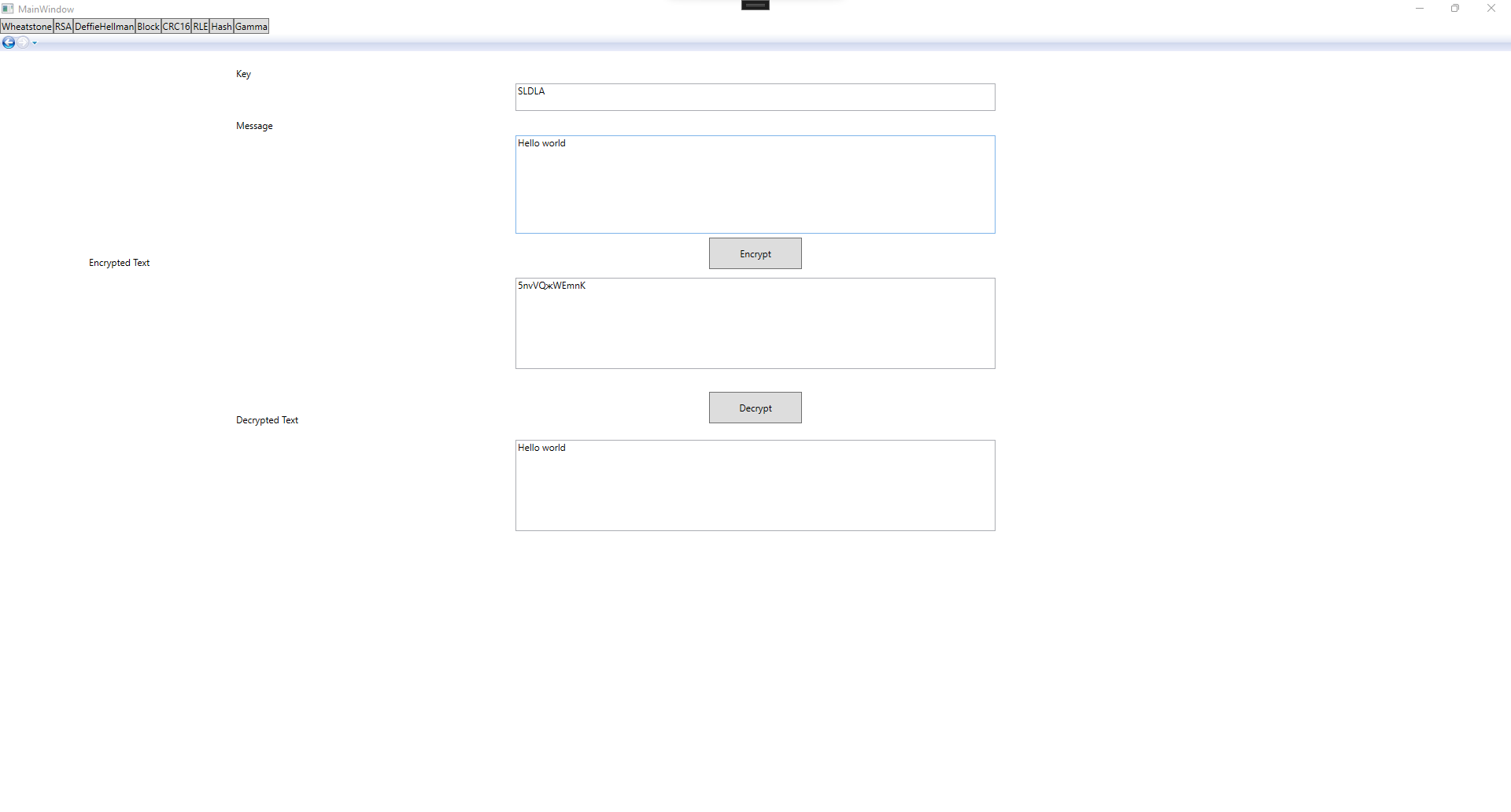


Рисунок 2 – Пример работы программы

**ПРИЛОЖЕНИЕ А**

Листинг A1 – Класс ApllicationViewModel

using GalaSoft.MvvmLight.Command;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Numerics;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

namespace Lab2

{

public class ApplicationViewModel : INotifyPropertyChanged

{

public event PropertyChangedEventHandler PropertyChanged;

private IFileService FileService;

private IDialogService DialogService;

private string text, encryptedText, decryptedText, key;

public ICommand OpenCommand { get; } *//команда открытия файла*

public ICommand EncryptCommand { get; }

public ICommand DecryptCommand { get; }

Crypto cr;

public ApplicationViewModel(IDialogService dialogServ, IFileService fileServ)

{

cr = new Crypto();

DialogService = dialogServ;

FileService = fileServ;

Key = "asdfghj";

OpenCommand = new RelayCommand(OpenFile);

EncryptCommand = new RelayCommand(Encrypt);

DecryptCommand = new RelayCommand(Decrypt);

}

public string Text

{

get => text;

set

{

text = value;

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("Text"));

}

}

public string EncryptedText

{

get => encryptedText;

set

{

encryptedText = value;

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("EncryptedText"));

}

}

public string DecryptedText

{

get => decryptedText;

set

{

decryptedText = value;

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("DecryptedText"));

}

}

public string Key

{

get => key;

set

{

key = value;

PropertyChanged?.Invoke(this, new PropertyChangedEventArgs("Key"));

}

}

private string alph = "1234567890qwertyuioplkjhgfdsazxcvbnmQWERTYUIOPLKJHGFDSAZXCVBNMёйцукенгшщзхъждлорпавыфячсмитьбюЁЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЮБЬТИМСЧЯ ";

private List<int> \_codes(string data)

{

List<int> codes = new List<int>();

foreach (var b in data.ToCharArray())

{

codes.Add(alph.IndexOf(b));

}

return codes;

}

private string \_str(List<int> codes)

{

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

foreach (var c in codes)

{

data.Add(alph[c]);

}

return String.Concat<char>(data);

}

private string encrypt(string data, string key)

{

var codes = \_codes(data);

var KeyCodes = \_codes(key);

List<int> eCodes = new List<int>();

for (int i = 0; i < codes.Count; i++) eCodes.Add(0);

for (var i = 0; i < codes.Count; i++)

{

eCodes[i] = (codes[i] ^ KeyCodes[i % KeyCodes.Count]);

}

return \_str(eCodes);

}

*//открытие файла*

private async void OpenFile()

{

if (DialogService.OpenFileDialog())

{

Text = await FileService.OpenAsync(DialogService.FilePath);

}

}

*//шифровка сообщения*

private void Encrypt()

{

EncryptedText = encrypt(Text, Key);

}

*//расшифровка сообщения*

private void Decrypt()

{

DecryptedText = encrypt(EncryptedText, Key);

}

}

}