Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Национальный исследовательский университет «МЭИ»

|  |  |
| --- | --- |
| Институт | ИВТИ |
| Кафедра | ПМИИ |

Дисциплина: «Защита данных»

Отчет по курсовой работе

«Разработка программы скрытия и извлечения информации в PDF файлах»

Вариант №42

Выполнил студент группы А-13а-19

Самсонова М. А.

Преподаватель: Хорев П. Б.

2022г.

Оглавление

[Введение 3](#_Toc122008162)

[1. Цель курсовой работы 3](#_Toc122008165)

[2. Задачи курсовой работы 3](#_Toc122008166)

[3. Требования к курсовой работе 3](#_Toc122008167)

[Глава 1 Описание алгоритма 5](#_Toc122008168)

[1. Алгоритм скрытия информации в графических файлах 5](#_Toc122008169)

[2. Алгоритм скрытия информации в PDF файлах 7](#_Toc122008170)

[Глава 2 Результаты проектирования 9](#_Toc122008171)

[1. Пользовательский интерфейс 9](#_Toc122008172)

[2. Программная реализация 14](#_Toc122008173)

[2.1. Скрытие и извлечение в графических файлах 14](#_Toc122008174)

[2.2. Скрытие и извлечение в PDF-файлах 17](#_Toc122008175)

[Глава 3 Тестирование разработанной программы 18](#_Toc122008176)

[Тестовый пример №1 18](#_Toc122008177)

[Тестовый пример №2 20](#_Toc122008178)

[Тестовый пример №3 22](#_Toc122008179)

[Тестовый пример №4 24](#_Toc122008180)

[Тестовый пример №5 25](#_Toc122008181)

[Тестовый пример №6 26](#_Toc122008182)

[Заключение 28](#_Toc122008183)

[Список источников 29](#_Toc122008184)

[Исходный код программы 30](#_Toc122008185)

# Введение

## Цель курсовой работы

Целью курсовой работы является разработка программы скрытия и извлечения информации в графических и PDF файлах.

## Задачи курсовой работы

В процессе выполнения курсовой работы:

* Разрабатывается пользовательский интерфейс приложения;
* Реализовывается алгоритм скрытия и извлечения информации из графических и PDF файлов с помощью языка C#;
* Интеграция алгоритма в пользовательский интерфейс;
* Тестирование и отладка программы;
* Подготовка отчёта по курсовой работе.

## Требования к курсовой работе

Разрабатываемый алгоритм должен поддерживать:

* возможность выбора файла-контейнера;
* возможность выбора файла-сообщения произвольного типа или ввода текста скрываемого сообщения;
* контроль возможности скрытия сообщения в контейнере (сравнением их длин, например);
* возможность шифровать/расшифровывать внедряемое/извлекаемое сообщение на ключе, выводимом из парольной фразы;
* возможность определять при расшифровании извлекаемых из контейнера данных факт ввода неверной парольной фразы (например, путем добавления к данным перед их шифрованием и внедрением в контейнер сигнатуры – специальной строки символов – с проверкой ее наличия в расшифрованных данных и удалением из них в случае успешной проверки).

# Описание алгоритма

Сокрытие информации может осуществляться различными методами. Одним из них является стеганография, которая скрывает сам факт передачи информации.

Основными понятиями в данном методе являются:

* Сообщение – объект, существование и содержание которого должно быть скрыто;
* Контейнер – объект, в котором скрывается сообщение;
* Ключ – секретный ключ, нужный для сокрытия контейнера. Ключ не шифрует данные, а скрывает место их нахождения в контейнере.

Алгоритмы, описанные ниже, основываются на стеганографии в различных контейнерах.

## Алгоритм скрытия информации в графических файлах

Одним из самых популярных методов стеганографии в графических файлах является метод наименьшего значащего бита (Least Significant Bit, LSB). Он основан на ограниченности зрения человека, которое не позволяет ему распознать всю палитру цветов большинства изображений. Таким образом, младшие биты цветов могут быть заменены на другие без потери информации. Важным ограничением в данном случае является то, что скрываемое сообщение должно быть меньше изображения или такого же размера, как изображение, используемое в качестве контейнера.

Для реализации данного алгоритма были выбраны изображения в формате PNG(portable network graphics) с глубиной цвета 24 и 32 бита (реализация на менее «глубоких» цветах делает невозможным скрытие больших объемов информации без потери качества изображения).

Изображения с глубиной цвета 24 бита состоят из пикселей, которые определяются 3 цветами RGB(R – red (красный), G – green (зелёный), B – blue (синий)), каждый из которых представляет 8 бит.

Изображения с глубиной цвета 32 бита состоят из пикселей, которые определяются 3 цветами RGB и альфа-каналом, каждый из которых представляет 8 бит.

Таким образом, изображение-контейнер имеет H\*W\*D\*8 бит информации, а для стеганографии в изображениях понадобится 2 младших бита изображения, которые позволят скрыть информацию объемом H\*W\*D\*2, где

* H – высота изображения;
* W – ширина изображения;
* D – глубина цвета изображения (D=3 если глубина 24 бита и D=4 если глубина 32 бита).

В алгоритме программы не будем использовать альфа-канал для скрытия информации и получим допустимый объем скрытия информации равный H\*W\*3\*2.

Схематично данный процесс изображен на рисунке 1.1.

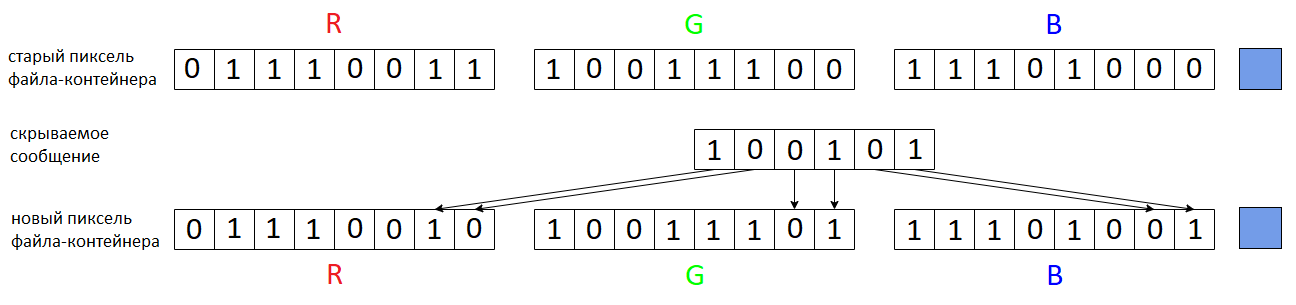


Рисунок . - Схема скрытия информации в один пиксель изображения

## Алгоритм скрытия информации в PDF файлах

Формат PDF - переносимый формат документов, разработанный компанией Adobe Systems. Структуру файла можно представить в виде 4 основных компонентов: однострочный заголовок, тело, таблица перекрестных ссылок и завершающий элемент.

Однострочный заголовок определяет версию спецификации PDF и имеет вид: %PDF–1.N, где N – версия от 0 до 7.

Тело файла PDF состоит из последовательности косвенных объектов, представляющих содержимое документа. Объекты представляют компоненты документа, например, шрифты, страницы и изображения.

Таблица перекрестных ссылок содержит информацию, которая предоставляет произвольный доступ к косвенным объектам в файле, так что нет необходимости читать весь файл, чтобы найти какой-либо конкретный объект. Таблица должна содержать однострочную запись для каждого косвенного объекта, указывающую смещение в байтах этого объекта в теле файла. Каждый раздел перекрестных ссылок должен начинаться со строки, содержащей ключевое слово xref.

Завершающий элемент указывает расположение таблицы перекрестных ссылок и некоторых специальных объектов в теле файла.

Скрывать информацию будем с помощью добавления нового объекта в структуру PDF-файла, при этом он не будет отображаться при чтении файла.

Алгоритм скрытия:

* Поиск и получение таблицы перекрестных ссылок с помощью ключевого слова xref;
* Получение скрываемого сообщения, вычисление его длины и определение номера объекта через таблицу перекрестных ссылок
* Добавление нового объекта по шаблону:

[N] 0 obj

<</Length [L] >>

stream

[text]

endstream

endobj

где [N] – номер объекта, [L] – длина сообщения, [text] – текст сообщения

* Обновление таблицы перекрестных ссылок

# Результаты проектирования

Приложение разработано для двух типов скрытия и извлечения информации, поэтому при открытии приложения предоставляет пользователю выбор вида стеганографии (Рисунок 2.1).

Выбор режима работы влияет на текст и на выбираемые файлы в интерфейсе программы.

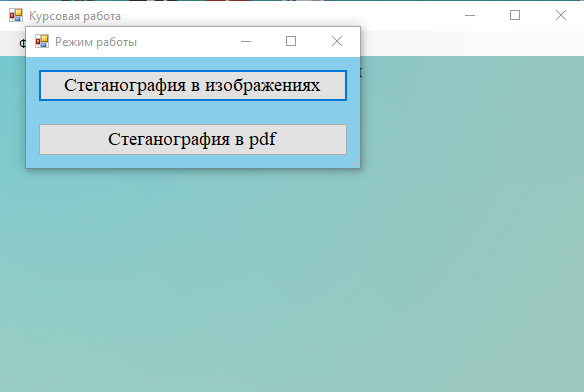


Рисунок . – Окно режима работы программы

## Пользовательский интерфейс

Независимо от выбора пользователь начинает работу с главным окном программы (Рисунок 2.2).

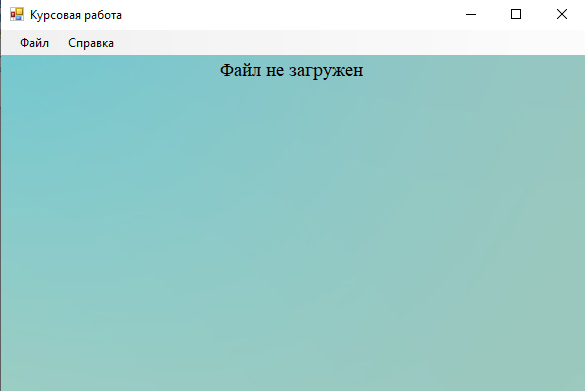


Рисунок . - Главное окно программы

Меню главного окна содержит разделы «Файл» и «Справка».

Раздел меню «Файл» позволяет выбрать файл-контейнер («Выбрать файл») или закрыть программу («Выход») (Рисунок 2.3, Рисунок 2.4).

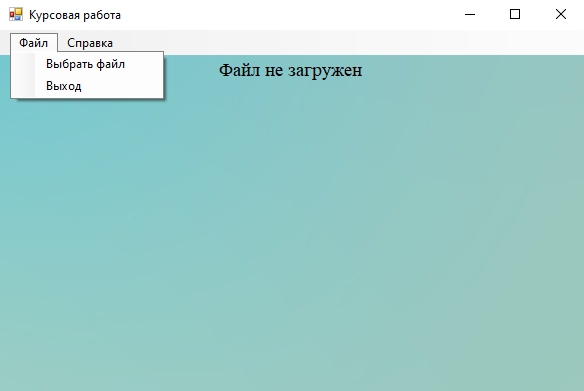


Рисунок . - Меню главного окна программы

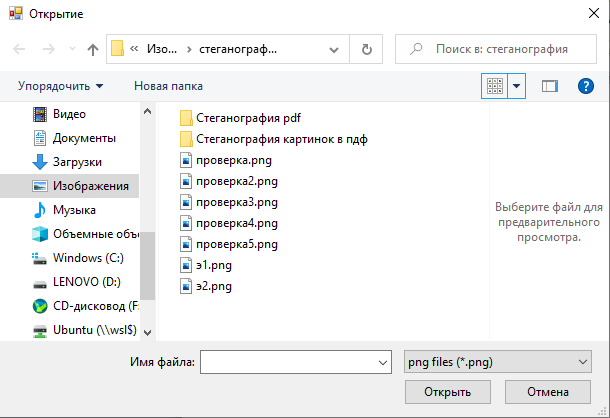


Рисунок . - Окно выбора файла-контейнера

Раздел меню «Справка» позволяет ознакомиться со сведениями программы («О программе») (Рисунок 2.5).

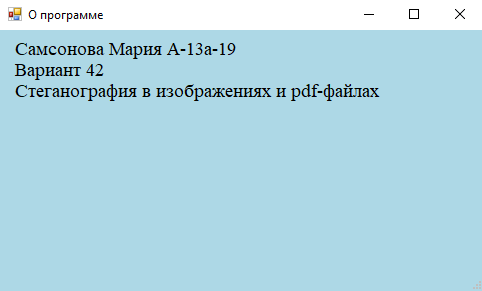


Рисунок . - Окно "О программе"

При успешном выборе файла происходит обновление главного окна (Рисунок 2.6).

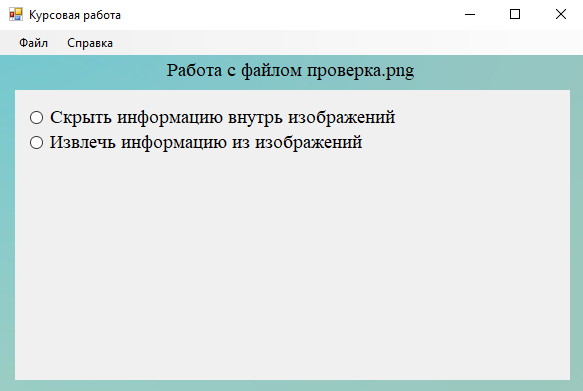


Рисунок . - Главное окно программы при успешном выборе файла-контейнера

При выборе типа действия «Скрытия» или «Извлечения» данных, происходит обновление окна в соответствии с выбранным пунктом (Рисунок 2.7, Рисунок 2.8).

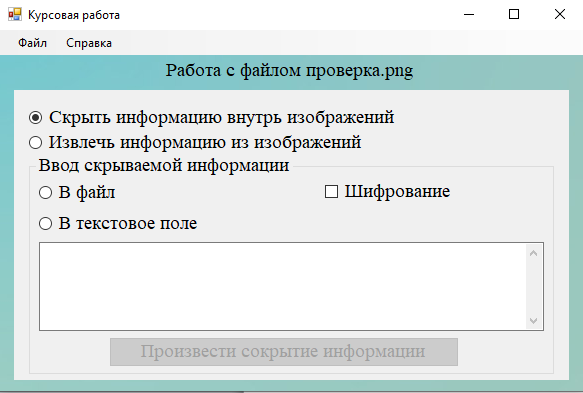


Рисунок . - Главное окно программы при выборе типа действия "Скрыть информацию внутрь изображений"

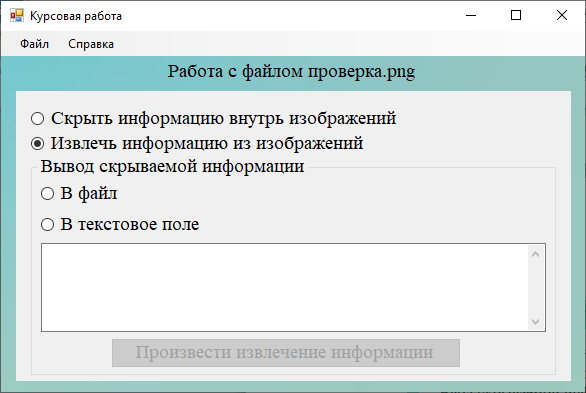


Рисунок . - Главное окно программы при выборе типа действия "Извлечь информацию внутрь изображений"

Нажатие на Checkbox «Шифрование» открывается окно ввода пароля пользователя для шифрования (Рисунок 2.9).

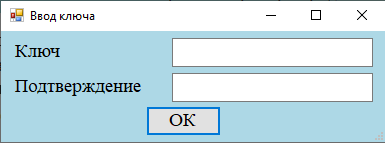


Рисунок . - Окно ввода пароля пользователя

Выбор RadioButton «В файл» предоставляет пользователю выбор скрываемого сообщения в форме файла формата TXT и PNG (Рисунок 2.10).

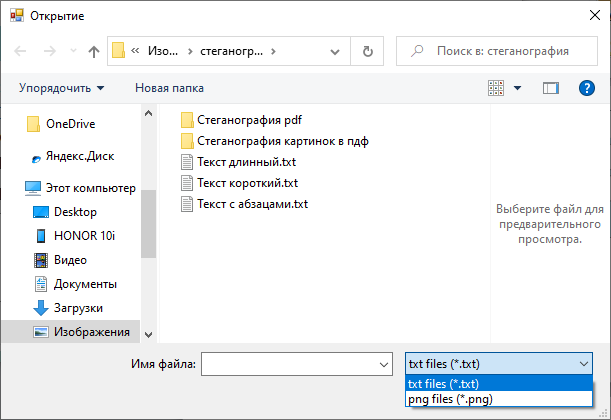


Рисунок . - Окно выбора скрываемого сообщения

При возникновении ошибок программа сигнализирует об этом пользователю (Рисунок 2.11, Рисунок 2.12, Рисунок 2.13, Рисунок 2.14, Рисунок 2.15).

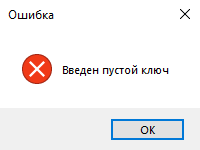


Рисунок . - Окно ошибки при вводе пустого пароля пользователя

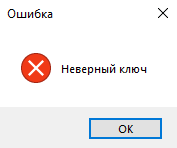


Рисунок . - Окно ошибки при вводе неверной парольной фразы

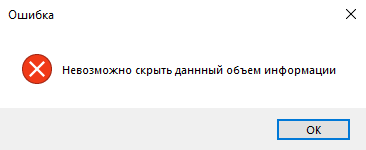


Рисунок . - Окно ошибки при невозможности скрытия информации данного объема

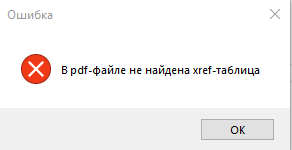


Рисунок . - Окно ошибки при отсутствии xref-таблицы подходящего типа

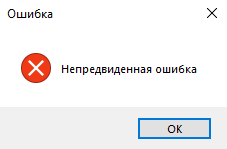


Рисунок . - Окно ошибки при обнаружении непредвиденного исключения

## Программная реализация

### Скрытие и извлечение в графических файлах

* Класс ImageSteg

Для реализации скрытия и извлечения сообщения из изображения был создан класс ImageSteg, который имеет поля:

* Bitmap image – исходный контейнер;
* Bitmap newimage – преобразованный контейнер;
* int salt – длина соли (примеси);
* string depth – глубина изображения;

Конструктор класса ImageSteg(Image img) – инициализирует объект класса, исходя из полученного объекта типа Image

Метод класса SaltInf() – возвращает длину соли (примеси)

Метод класса SizeImage() – возвращает размер изображения в байтах

Метод класса TextToImage(byte[] bytestext) – реализует скрытие сообщения в контейнер

Метод класса ImageToText() – реализует извлечение сообщения из контейнера

* Класс Form1

Поля:

* bool stegimageorpdf – режим работы программы (false – изображения, true – pdf-файлы);
* string filename – путь к файлу контейнеру;
* string name – имя файла контейнера;
* Bitmap fileimage – изображение-контейнер;
* string filenamesource = "" – путь к скрываемому файлу;
* bool textorimage – тип скрываемого сообщения (0 - текст, 1 - картинка);
* string textsource = "" – текст скрываемого сообщения;
* string enctextsource = "" – зашифрованный текст скрываемого сообщения;
* long maxsizeinf – максимальная длина сообщения в битах;
* long sizeinf – длина сообщения в битах;
* string key = "" – ключ пользователя для шифрования;
* bool mode – режим работы(true - скрытие ,false - извлечение);
* TripleDESCryptoServiceProvider DES3 – объект класса для криптоалгоритма;
* MD4CryptoServiceProvider MD4 – объект класса хеширования;
* RNGCryptoServiceProvider rand – создание объекта для генерации случайной примеси;
* byte[] randBytes – буфер для случайной примеси;
* CryptoStream crypts – объект для криптографического потока данных;
* MemoryStream mems – объект класса для потока данных в оперативной памяти;
* byte[] pwd – пароль пользователя;
* byte[] IV – начальный вектор;

Обработка событий:

* выходToolStripMenuItem\_Click(object sender, EventArgs e) – выход из программы при нажатии на соответствующий раздел меню
* выбратьФайлToolStripMenuItem\_Click(object sender, EventArgs e) – выбор файла-контейнера программы при нажатии на соответствующий раздел меню
* оПрограммеToolStripMenuItem\_Click(object sender, EventArgs e) – показ формы «О программе» при нажатии на соответствующий раздел меню
* RBStegPict\_CheckedChanged(object sender, EventArgs e) – изменение отображения главной формы при изменении значения элемента «Скрыть информацию внутрь изображений»
* RBExtPict\_CheckedChanged(object sender, EventArgs e) – изменение отображения главной формы при изменении значения элемента «Извлечь информацию из изображений»
* RBInfFile\_CheckedChanged(object sender, EventArgs e) – изменение отображения главной формы при изменении значения элемента «В файл»
* RBInfText\_CheckedChanged(object sender, EventArgs e) – изменение отображения главной формы при изменении значения элемента «В текстовое поле»
* ButProc\_Click(object sender, EventArgs e) – вызов функций для реализации алгоритма скрытия или извлечения информации с предпроверкой на допустимый объем информации
* CBEnc\_CheckedChanged(object sender, EventArgs e) – показ формы ввода ключа для шифрованию при изменении значения элемента «Шифрование»
* Form1\_Shown(object sender, EventArgs e) – подготовка программы к использованию при первом открытии главной формы

Функция EncText(string text, int sizerand) – реализует шифрование сообщения text с помощью алгоритма 3DES на ключе, построенном с помощью MD4 на основе парольной фразы.

Функция DecText(byte[] bytestextsource, int sizerand) – реализует дешифрование сообщения bytestextsource.

* Класс KeyForm

Реализация формы ввода парольной фразы для шифрования или дешифрования

* Класс AboutForm

Реализация формы «О программе»

* Класс SelectFun

Реализация формы выбора режима работы программы

### Скрытие и извлечение в PDF-файлах

* Класс Form1

Аналогичен пункту 2.1 +

Поля:

* string[,] xreftable – таблица перекрестных ссылок файла;
* int numobj – номер добавляемого объекта;

Функция PdfStegFun(string filename, byte[] text, int num) – скрывает сообщение text в pdf-файл

Функция PdfEncFun(string filename, int num) – извлекает сообщение из pdf-файла

* Класс KeyForm

Аналогичен пункту 2.1

* Класс AboutForm

Аналогичен пункту 2.1

* Класс SelectFun

Аналогичен пункту 2.1

# Тестирование разработанной программы

Для тестирования интерфейса приложения и алгоритмов скрытия и извлечения информации опишем тестовые примеры и сравним их результаты с ожидаемыми данными.

## Тестовый пример №1

Скрытие и извлечение информации в 24битном изображении с использованием текстового поля без шифрования.

Исходный файл-контейнер – Nats.png

Сообщение – «СКРЫТИЕ ТЕКСТА БЕЗ ШИФРОВАНИЯ»

Полученный файл-контейнер – 1.png



Рисунок . - Файл-контейнер Nats.png

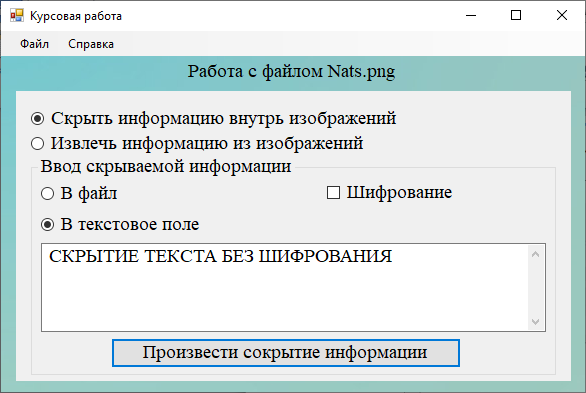


Рисунок . - Скрытие информации в 24битном изображений без использования шифрования



Рисунок . - Файл-контейнер 1.png

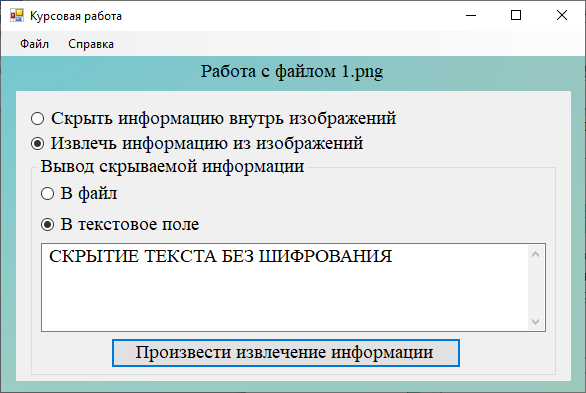


Рисунок . - Извлечение информации в 24битном изображений без использования шифрования

## Тестовый пример №2

Скрытие и извлечение информации в 24битном изображении с использованием файла в формате TXT и шифрованием.

Исходный файл-контейнер – Nats.png

Сообщение – Текст длинный.txt

Полученный файл-контейнер – 2.png

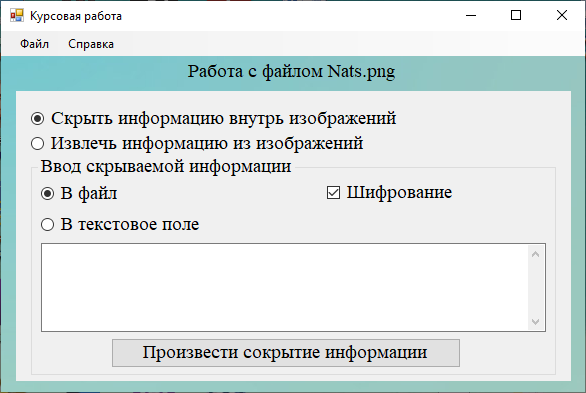


Рисунок . - Скрытие информации в 24битном изображений с использованием шифрования и файла-сообщения типа TXT

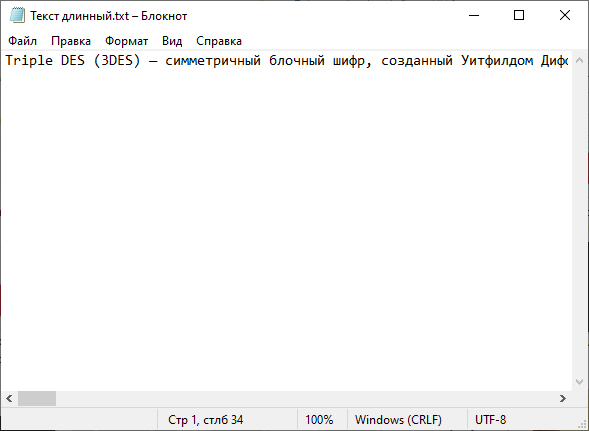


Рисунок . - Файл-сообщение Текст длинный.txt



Рисунок . - Файл-контейнер 2.png

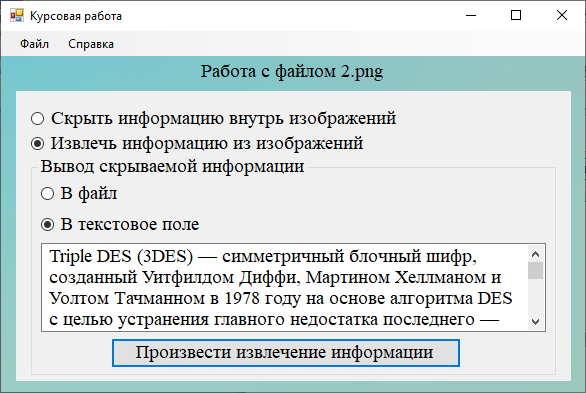


Рисунок . - Извлечение информации в 24битном изображений с использованием шифрования и файла-сообщения типа TXT

## Тестовый пример №3

Скрытие и извлечение информации в 24битном изображении с использованием файла в формате PNG и шифрования.

Исходный файл-контейнер – Nats.png

Сообщение – 4\_3.png

Полученный файл-контейнер – 3.png

Извлеченное сообщение – 3(1).png

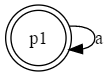


Рисунок . - Файл-сообщение 4\_3.png

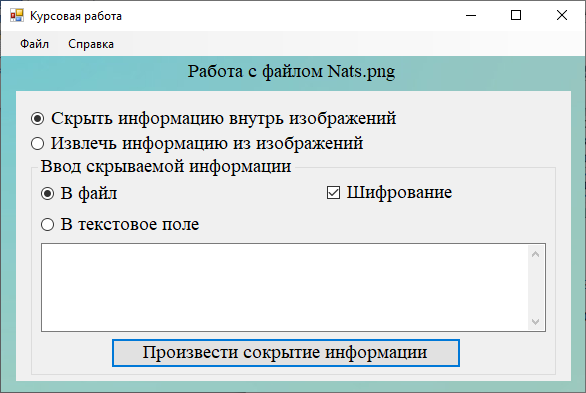


Рисунок . - Скрытие информации в 24битном изображении с использованием файла в формате PNG и шифрованием

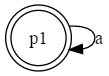


Рисунок . - Извлеченное сообщение 3(1).png

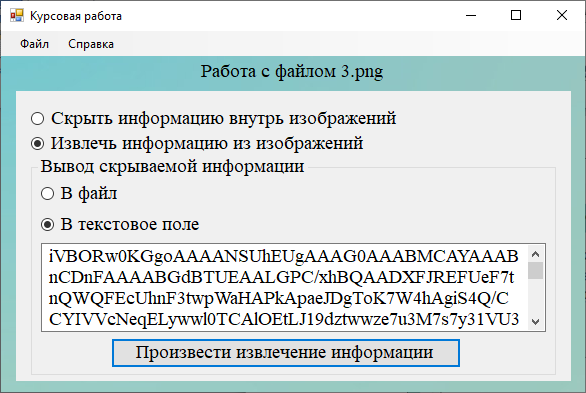


Рисунок . - Извлечение информации в 24битном изображении с использованием файла в формате PNG и шифрованием

## Тестовый пример №4

Скрытие и извлечение информации в PDF-файле с использованием текстового поля без шифрования.

Исходный файл-контейнер – PDF с картинкой.pdf

Сообщение – «СКРЫТИЕ БЕЗ ШИФРОВАНИЯ»

Полученный файл-контейнер – 1.pdf

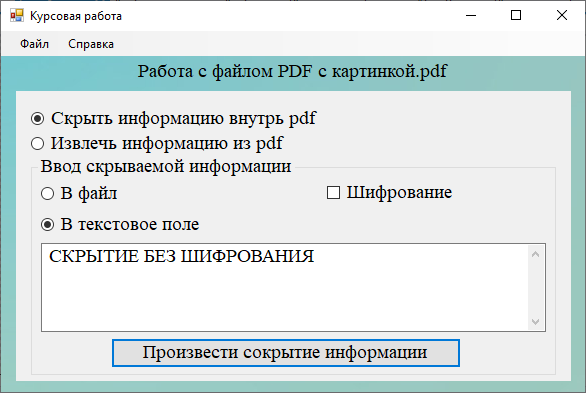


Рисунок . – Скрытие информации в PDF-файле с использованием текстового поля без шифрования

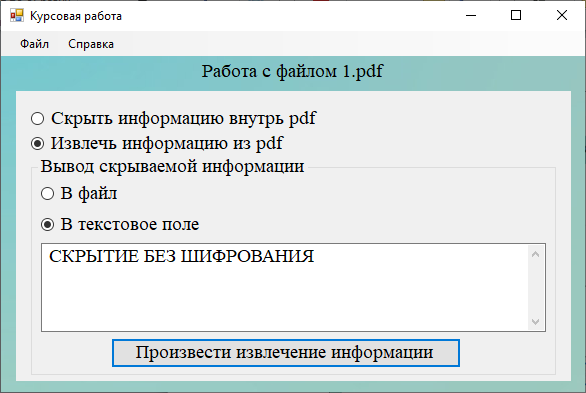


Рисунок . - Скрытие и извлечение информации в PDF-файле с использованием текстового поля без шифрования

## Тестовый пример №5

Скрытие и извлечение информации в PDF-файле с использованием текстового поля c шифрованием.

Исходный файл-контейнер – PDF Вопросы к экзамену.pdf

Сообщение – «СКРЫТИЕ ТЕКСТА С ШИФРОВАНИЕМ»

Полученный файл-контейнер – 2.pdf

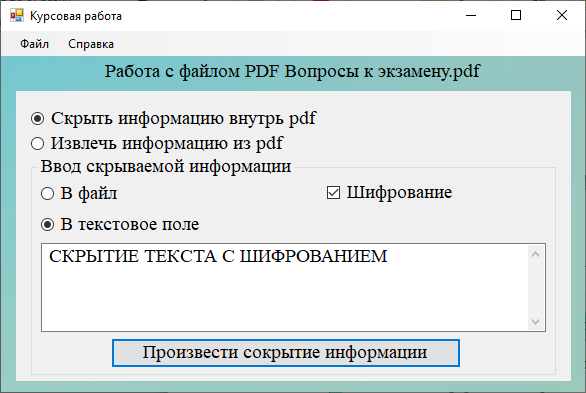


Рисунок . – Скрытие информации в PDF-файле с использованием текстового поля c шифрованием

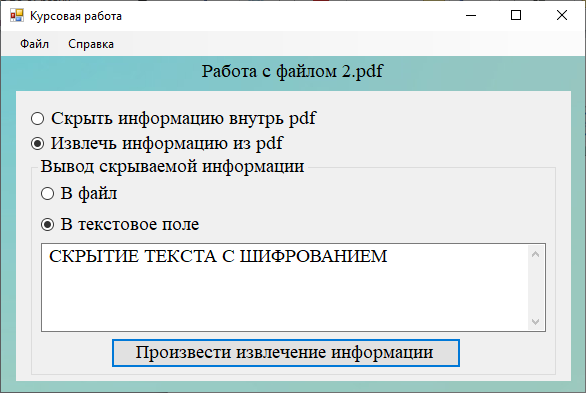


Рисунок . - Извлечение информации в PDF-файле с использованием текстового поля c шифрованием

## Тестовый пример №6

Скрытие и извлечение информации в PDF-файле с использованием файла-сообщения в формате PNG без шифрования.

Исходный файл-контейнер – PDF Вопросы к экзамену.pdf

Сообщение – Проверка.png

Полученный файл-контейнер – 3.pdf

Извлеченное сообщение – 3(1).png

C:\Users\Мария\Pictures\стеганография\проверка.png

Рисунок . - Файл-сообщение Проверка.png

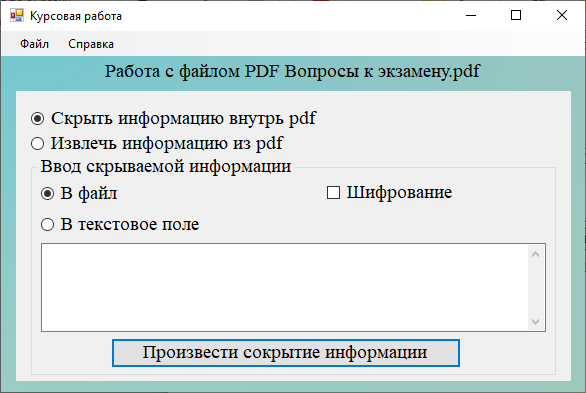


Рисунок . - Скрытие информации в PDF-файле с использованием файла-сообщения в формате PNG без шифрования

C:\Users\Мария\Pictures\стеганография\Стеганография pdf\3(1).png

Рисунок . - Извлеченное сообщение 3(1).png

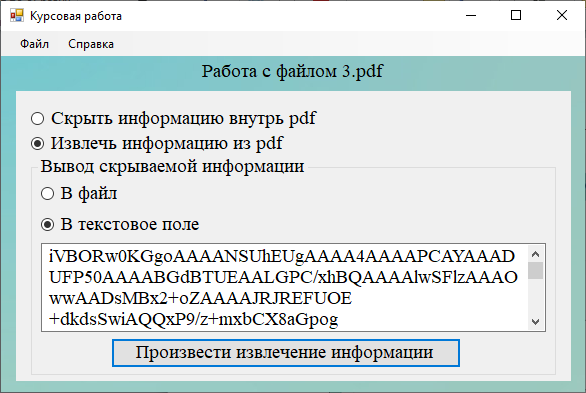


Рисунок . - Извлечение информации в PDF-файле с использованием файла-сообщения в формате PNG без шифрования

# Заключение

В ходе выполнения курсовой работы по заданной теме:

* Изучен и реализован метод LSB(Least Significant Bit) для скрытия сообщения внутри изображений
* Изучено устройство PDF-файлов и реализовано скрытие информации в нем с помощью добавления неиспользуемого при отображении объекта файла
* Спроектирован и разработан интерфейс приложения с помощью языка C# и платформы Windows Forms
* Произведено тестирование приложения по тестовым примерам
* Составлен отчет по курсовой работе

# Список источников

1. Dr. Ekta Walia, Payal Jain, Navdeep, “An Analysis of LSB & DCT based Steganography”, Global Journal of Computer Science and Technology Vol. 10 Issue 1 (Ver 1.0), April 2010, Page 4-8.
2. Грибунин В. Г., Оков И. Н., Туринцев И. В., Цифровая стеганография: Солон-Пресс, 2009. – 272 с.
3. Adobe Systems Incorporated, Portable Document Format Reference Manual, version 1.7, November 2006.
4. Формат PDF

URL: http://el-izdanie.narod.ru/gl3/3-2.htm

# Исходный код программы

Form1.cs

using System;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Security.Cryptography;

using Org.Mentalis.Security.Cryptography;

using System.IO;

using System.Drawing.Imaging;

namespace КР

{

public partial class Form1 : Form

{

public bool stegimageorpdf;

private string filename; //путь к файлу контейнеру

private string name; //имя файла контейнера

private string[,] xreftable; //таблица перекрестных ссылок файла

private int numobj;

private Bitmap fileimage;

private string filenamesource = ""; //путь к скрываемому файлу

private bool textorimage; //тип скрываемого сообщения (0 - текст, 1 - картинка)

private string textsource = ""; //текст скрываемого сообщения

private string enctextsource = ""; //зашифрованный текст скрываемого сообщения

private long maxsizeinf; //максимальная длина сообщения в битах

private long sizeinf; //длина сообщения в битах

public string key = ""; //ключ пользователя для шифрования

public bool mode; //режим работы(true - скрытие ,false - извлечение)

//Криптография

private TripleDESCryptoServiceProvider DES3; // объект класса для криптоалгоритма

private MD4CryptoServiceProvider MD4; // объект класса хеширования

private RNGCryptoServiceProvider rand; // создание объекта для генерации случайной примеси

private byte[] randBytes; // буфер для случайной примеси

private CryptoStream crypts; //объект для криптографического потока данных

private MemoryStream mems; // объект класса для потока данных в оперативной памяти

private byte[] pwd; // пароль пользователя

private byte[] IV; // начальный вектор.

public byte[] EncText(string text, int sizerand) //шифрование сообщения

{

text = "ШИФРОВАНИЕ " + text;

if(!stegimageorpdf)

pwd = Encoding.Unicode.GetBytes(key);

else

pwd = Encoding.Default.GetBytes(key);

randBytes = new byte[sizerand];

rand.GetBytes(randBytes); // соль

byte[] pwds = new byte[pwd.Length + randBytes.Length]; // пароль + соль

// добавление к паролю пользователя соли

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

{

if (i < pwd.Length)

pwds[i] = pwd[i];

else

pwds[i] = randBytes[i - pwd.Length];

}

if (!stegimageorpdf)

pwds = Encoding.Unicode.GetBytes(Encoding.Unicode.GetString(pwds));

else

pwds = Encoding.Default.GetBytes(Encoding.Default.GetString(pwds));

DES3.Key = Convert.FromBase64String(Convert.ToBase64String(MD4.ComputeHash(pwds))); // генерация ключа шифрования

ICryptoTransform encryptor = DES3.CreateEncryptor(DES3.Key, IV); // создание объекта шифрования

while (text.Length % 8 != 0)

{

text = text + '0';

}

byte[] fullfileb;

if (!stegimageorpdf)

fullfileb = Encoding.Unicode.GetBytes(text);

else

fullfileb = Encoding.Default.GetBytes(text);

mems = new MemoryStream();

crypts = new CryptoStream(mems, encryptor, CryptoStreamMode.Write);

crypts.Write(fullfileb, 0, fullfileb.Length);

crypts.FlushFinalBlock();

crypts.Close();

mems.Flush();

byte[] encfile = mems.ToArray();

DES3.Clear();

mems.Close();

byte[] bytestextsource = new byte[randBytes.Length + encfile.Length];

// добавление к файлу пользователей соли

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

{

if (i < randBytes.Length)

bytestextsource[i] = randBytes[i];

else

bytestextsource[i] = encfile[i - randBytes.Length];

}

return bytestextsource;

}

public byte[] DecText(byte[] bytestextsource, int sizerand)

{

byte[] file = bytestextsource;

randBytes = new byte[sizerand];

byte[] defile = new byte[file.Length - sizerand];

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

{

if (i < sizerand)

randBytes[i] = file[i];

else

defile[i - sizerand] = file[i];

}

byte[] pwds = new byte[pwd.Length + randBytes.Length]; // пароль + соль

// добавление к паролю пользователя соли

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

{

if (i < pwd.Length)

pwds[i] = pwd[i];

else

pwds[i] = randBytes[i - pwd.Length];

}

if (!stegimageorpdf)

pwds = Encoding.Unicode.GetBytes(Encoding.Unicode.GetString(pwds));

else

pwds = Encoding.Default.GetBytes(Encoding.Default.GetString(pwds));

DES3.Key = Convert.FromBase64String(Convert.ToBase64String(MD4.ComputeHash(pwds))); // генерация ключа шифрования

ICryptoTransform decryptor = DES3.CreateDecryptor(DES3.Key, IV);

mems = new MemoryStream(defile);

crypts = new CryptoStream(mems, decryptor, CryptoStreamMode.Read);

byte[] toEncrypt = new byte[defile.Length];

Console.WriteLine(defile.Length);

crypts.Read(toEncrypt, 0, toEncrypt.Length);

crypts.Flush();

crypts.Close();

mems.Flush();

mems.Close();

return toEncrypt;

}

public string[,] XrefTable(string filename)

{

string[,] values = new string[0, 3];

int kol;

int firstobj = 0;

int numxref = 0;

int i = 0;

bool endxref = false;

bool isstream = false;

string text1 = File.ReadAllText(filename, Encoding.Default);

string[] alllines = text1.Split('\n');

foreach (string s in alllines)

{

if (!isstream && s.Contains("xref"))

{

numxref = i;

}

else if (!endxref && numxref != 0)

{

if (s.Contains("trailer"))

{

endxref = true;

}

else if (s.Length < 18)

{

string[] subs = s.Split(' ');

firstobj = Convert.ToInt32(subs[0]);

kol = Convert.ToInt32(subs[1]);

values = new string[firstobj + kol, 3];

}

else if (s.Length >= 18)

{

string[] subs = s.Split(' ');

values[firstobj + i - numxref - 2, 0] = subs[0];

values[firstobj + i - numxref - 2, 1] = subs[1];

values[firstobj + i - numxref - 2, 2] = subs[2];

}

}

if (s.Contains("stream"))

isstream = true;

else

isstream = false;

i++;

}

return values;

}

public string PdfStegFun(string filename, byte[] text, int num)

{

string stext = Encoding.Default.GetString(text);

string newobj;

newobj = num.ToString() + " 0 obj\n<</Length " + stext.Length + ">> \nstream\n" + stext + "\nendstream\n" + "endobj \n";

int offsetobj = newobj.Length + (num+1).ToString().Length - num.ToString().Length;

string newxrefstr = "";

string oldsizexrefstr = "";

string newsizexrefstr = "";

string newstartxreffile = "";

string startfile = "";

string xreffile = "";

string middlefile = "";

string startxreffile = "";

string endfile = "";

int i = 0;

bool start = true;

bool startxref = false;

bool endxref = false;

bool startstartxref = false;

bool endstartxref = false;

string text1 = File.ReadAllText(filename, Encoding.Default);

string[] alllines = text1.Split('\n');

foreach (string s in alllines)

{

if (s.Contains("xref"))

{

start = false;

startxref = true;

}

if (s.Contains("trailer"))

{

endxref = true;

}

if (s.Contains("startxref"))

{

startstartxref = true;

}

if (start)

{

startfile += s + '\n';

}

else if (!endxref && startxref)

{

if (!s.Contains("xref"))

{

if (s.Length < 18)

oldsizexrefstr = s;

else

xreffile += s + '\n';

}

}

else if (!endstartxref && startstartxref)

{

if (!s.Contains("startxref"))

{

startxreffile += s;

endstartxref = true;

}

}

else if (!start && endxref && !startstartxref && !endstartxref)

{

middlefile += s + '\n';

}

else

{

endfile += s + '\n';

}

i++;

}

string[] substr1 = oldsizexrefstr.Split(' ');

newsizexrefstr = "xref\n" + substr1[0] + " " + ((numobj + 1)-Convert.ToInt32(substr1[0])).ToString() + "\n";

string[] substr2 = xreffile.Split('\n');

string str2 = substr2[substr2.Length - 2];

string[] substr3 = str2.Split(' ');

string str3 = substr3[0];

int newoffset = Convert.ToInt32(str3) + offsetobj;

str3 = newoffset.ToString();

while (str3.Length < 10)

str3 = "0" + str3;

newstartxreffile = (Convert.ToInt32(startxreffile) + offsetobj + newsizexrefstr.Length - 5).ToString() + "\n";

newxrefstr = str3 + " " + substr3[1] + " " + substr3[2] + "\n";

string newfile = startfile + newobj + newsizexrefstr + xreffile + newxrefstr + middlefile + "startxref\n"+ newstartxreffile + endfile;

return newfile;

}

public string PdfEncFun(string filename, int num)

{

string text1 = File.ReadAllText(filename, Encoding.Default);

string[] alllines = text1.Split('\n');

string obj = num.ToString() + " 0 obj";

bool startobj = false;

int i = 0;

string text = "";

foreach (string s in alllines)

{

if (s.Contains(obj))

{

startobj = true;

}

if(s.Contains("endstream"))

{

startobj = false;

}

if(startobj)

{

if(i > 2)

{

text += s;

}

i++;

}

}

return text;

}

public Form1()

{

InitializeComponent();

MainPanel.Visible = false;

MainGB.Visible = false;

}

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

{

Close();

}

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

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

if(!stegimageorpdf)

{

openFileDialog1.Filter = "png files (\*.png)|\*.png";

openFileDialog1.RestoreDirectory = false;

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

filename = openFileDialog1.FileName;

var fi1 = new FileInfo(filename);

name = fi1.Name;

fileimage = new Bitmap(filename);

RBStegPict.Text = "Скрыть информацию внутрь изображений";

RBExtPict.Text = "Извлечь информацию из изображений";

MainPanel.Visible = true;

}

}

else

{

openFileDialog1.Filter = "pdf files (\*.pdf)|\*.pdf";

openFileDialog1.RestoreDirectory = false;

if (openFileDialog1.ShowDialog() == DialogResult.OK)

{

filename = openFileDialog1.FileName;

var fi1 = new FileInfo(filename);

name = fi1.Name;

try

{

xreftable = XrefTable(filename);

numobj = xreftable.GetUpperBound(0) + 1;

RBStegPict.Text = "Скрыть информацию внутрь pdf";

RBExtPict.Text = "Извлечь информацию из pdf";

MainPanel.Visible = true;

}

catch

{

MessageBox.Show("В pdf-файле не найдена xref-таблица", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

/\* неудачная попытка стеганографии картинки в пдф

for (int i = 0; i <= xreftable.GetUpperBound(0); i++)

{

if (xreftable[i, 1] != "65535" && IsImage(filestring, Convert.ToInt32(xreftable[i, 0])))

{

offsetobj = Convert.ToInt32(xreftable[i, 0]);

numobj = i;

break;

}

}

if (numobj < xreftable.GetUpperBound(0))

{

fileimage = ExtractImage(xreftable, filestring, offsetobj);

MainPanel.Visible = true;

}

else

{

MainPanel.Visible = false;

}

\*/

}

}

LabelNameFile.Text = "Работа с файлом " + name;

}

private void RBStegPict\_CheckedChanged(object sender, EventArgs e)//изменение рб скрытия в изображения

{

if (RBStegPict.Checked)

{

MainGB.Visible = true;

RBInfFile.Checked = false;

RBInfText.Checked = false;

MainGB.Text = "Ввод скрываемой информации";

ButProc.Text = "Произвести сокрытие информации";

ButProc.Enabled = false;

TBTypeText.Text = "";

CBEnc.Visible = true;

CBEnc.Checked = false;

mode = true;

}

}

private void RBInfFile\_CheckedChanged(object sender, EventArgs e)//изменение рб ввода/вывода через файл

{

if (RBInfFile.Checked)

{

if(RBStegPict.Checked)

{

OpenFileDialog openFileDialog1 = new OpenFileDialog();

openFileDialog1.Filter = "txt files (\*.txt)|\*.txt|png files (\*.png)|\*.png";

openFileDialog1.RestoreDirectory = false;

if (openFileDialog1.ShowDialog() == DialogResult.OK)

filenamesource = openFileDialog1.FileName;

textorimage = Path.GetExtension(filenamesource) == ".png";

}

if (RBStegPict.Checked && filenamesource != "" || !RBStegPict.Checked)

{

ButProc.Enabled = true;

}

}

}

private void RBInfText\_CheckedChanged(object sender, EventArgs e)//изменение рб ввода/вывода через тб

{

if (RBInfText.Checked)

{

textorimage = false;

ButProc.Enabled = true;

}

else

{

ButProc.Enabled = false;

}

}

private void RBExtPict\_CheckedChanged(object sender, EventArgs e)//изменение рб извлечения в изображения

{

if(RBExtPict.Checked)

{

MainGB.Visible = true;

RBInfFile.Checked = false;

RBInfText.Checked = false;

MainGB.Text = "Вывод скрываемой информации";

ButProc.Text = "Произвести извлечение информации";

ButProc.Enabled = false;

TBTypeText.Text = "";

CBEnc.Visible = false;

mode = false;

}

}

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

{

try

{

// работа с картинкой

if (!stegimageorpdf)

{

Image im = (Image)fileimage;

if (mode)

{

ImageSteg image = new ImageSteg(im);

maxsizeinf = image.SizeImage() \* 2;

Console.WriteLine("Максимальное кол-во бит информации " + maxsizeinf.ToString());

if (!textorimage)

{

//Получение скрываемого текста

if (RBInfText.Checked)

textsource = "СТЕГАНОГРАФИЯ " + TBTypeText.Text + " СТЕГАНОГРАФИЯ" + '\n';

else if (RBInfFile.Checked)

textsource = "СТЕГАНОГРАФИЯ " + File.ReadAllText(filenamesource) + " СТЕГАНОГРАФИЯ" + '\n';

}

else

{

Image imagesourse = Image.FromFile(filenamesource);

using (MemoryStream m = new MemoryStream())

{

imagesourse.Save(m, imagesourse.RawFormat);

byte[] imageBytes = m.ToArray();

textsource = Convert.ToBase64String(imageBytes);

}

Console.WriteLine("Картинка в байтах" + textsource.Length);

textsource = "КАРТИНКА СТЕГАНОГРАФИЯ " + textsource + " СТЕГАНОГРАФИЯ" + '\n';

}

byte[] bytestextsource;

//режим работы (с шифрованием или без)

if (CBEnc.Checked)

{

//Шифрование текста сообщения

int sizerand = image.SaltInf();

bytestextsource = EncText(textsource, sizerand);

}

else

{

bytestextsource = Encoding.Unicode.GetBytes(textsource);

}

sizeinf = bytestextsource.Length \* 8;

Console.WriteLine("Кол-во бит информации " + sizeinf.ToString());

if (maxsizeinf >= sizeinf)

{

//функция сокрытия

Bitmap newimage = image.TextToImage(bytestextsource);

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "png files (\*.png)|\*.png";

saveFileDialog1.RestoreDirectory = false;

if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

string savefilename = saveFileDialog1.FileName;

Bitmap newimageformat = new Bitmap(newimage.Width, newimage.Height, fileimage.PixelFormat);

using (Graphics gr = Graphics.FromImage(newimageformat))

{

gr.DrawImage(newimage, new Rectangle(0, 0, newimageformat.Width, newimageformat.Height));

}

newimageformat.Save(savefilename, ImageFormat.Png);

/\* неудачная попытка стеганографии в картинки пдф

Bitmap bm = new Bitmap("newimg.png");

byte[ ] data = new byte[3 \* bm.Height \* bm.Width];

for (int i = 0; i < bm.Height; i++)

{

for(int j = 0; j < bm.Width; j++)

{

Color color = bm.GetPixel(j, i);

data[i \* 3 \* bm.Width + j \* 3] = color.R;

data[i \* 3 \* bm.Width + j \* 3 + 1] = color.G;

data[i \* 3 \* bm.Width + j \* 3 + 2] = color.B;

}

}

byte[] newfile = InsertImage(filestring, offsetobj, numobj, data);

File.WriteAllBytes(savefilename, newfile);

\*/

}

else

{

MessageBox.Show("Невозможно скрыть даннный объем информации", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

if (!mode)

{

//функция извлечения

ImageSteg image = new ImageSteg(im);

byte[] bytestextsource = image.ImageToText();

Console.WriteLine(Encoding.Unicode.GetString(bytestextsource));

if (!Encoding.Unicode.GetString(bytestextsource).Contains("СТЕГАНОГРАФИЯ"))

{

KeyForm kf = new KeyForm();

kf.f1 = this;

kf.ShowDialog();

pwd = Encoding.Unicode.GetBytes(key);

//Расшифрование текста сообщения

int sizerand = image.SaltInf();

enctextsource = Encoding.Unicode.GetString(DecText(bytestextsource, sizerand));

if (enctextsource.Contains("ШИФРОВАНИЕ"))

{

textsource = enctextsource.Substring(11);

if (textsource.Contains("КАРТИНКА"))

{

textsource = textsource.Substring(9);

}

}

else

{

MessageBox.Show("Неверный ключ", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

else

{

textsource = Encoding.Unicode.GetString(bytestextsource);

if (Encoding.Unicode.GetString(bytestextsource).Contains("КАРТИНКА"))

{

textsource = textsource.Substring(9);

}

}

textsource = textsource.Substring(14);

Console.WriteLine(textsource);

textsource = textsource.Substring(0, textsource.LastIndexOf(" СТЕГАНОГРАФИЯ"));

if (RBInfText.Checked)

{

TBTypeText.Text = textsource;

}

else if (RBInfFile.Checked)

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "txt files (\*.txt)|\*.txt|png files (\*.png)|\*.png";

saveFileDialog1.RestoreDirectory = false;

if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

string savefilename = saveFileDialog1.FileName;

if (Path.GetExtension(savefilename) == ".png")

{

Console.WriteLine(textsource.Length);

byte[] data = Convert.FromBase64String(textsource);

using (var stream = new MemoryStream(data, 0, data.Length))

{

Image newimage = Image.FromStream(stream);

newimage.Save(savefilename, ImageFormat.Png);

}

}

else if (Path.GetExtension(savefilename) == ".txt")

{

StreamWriter writer = new StreamWriter(savefilename);

writer.Write(textsource);

writer.Dispose();

writer.Close();

}

}

}

}

// работа с пдф

else

{

if (mode)

{

maxsizeinf = 20000;

Console.WriteLine("Максимальное кол-во бит информации " + maxsizeinf.ToString());

if (!textorimage)

{

//Получение скрываемого текста

if (RBInfText.Checked)

textsource = "СТЕГАНОГРАФИЯ " + TBTypeText.Text + " СТЕГАНОГРАФИЯ" + '\n';

else if (RBInfFile.Checked)

textsource = "СТЕГАНОГРАФИЯ " + File.ReadAllText(filenamesource,Encoding.Default) + " СТЕГАНОГРАФИЯ" + '\n';

}

else

{

Image imagesourse = Image.FromFile(filenamesource);

using (MemoryStream m = new MemoryStream())

{

imagesourse.Save(m, imagesourse.RawFormat);

byte[] imageBytes = m.ToArray();

textsource = Convert.ToBase64String(imageBytes);

}

Console.WriteLine("Картинка в байтах" + textsource.Length);

textsource = "КАРТИНКА СТЕГАНОГРАФИЯ " + textsource + " СТЕГАНОГРАФИЯ" + '\n';

}

byte[] bytestextsource;

//режим работы (с шифрованием или без)

if (CBEnc.Checked)

{

//Шифрование текста сообщения

int sizerand = 8;

bytestextsource = EncText(textsource, sizerand);

}

else

{

bytestextsource = Encoding.Default.GetBytes(textsource);

}

Console.WriteLine("зашифрованный " + Encoding.Default.GetString(bytestextsource));

sizeinf = bytestextsource.Length \* 8;

Console.WriteLine("Кол-во бит информации " + sizeinf.ToString());

if (maxsizeinf >= sizeinf)

{

//функция сокрытия

string newfile = PdfStegFun(filename, bytestextsource, numobj);

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "pdf files (\*.pdf)|\*.pdf";

saveFileDialog1.RestoreDirectory = false;

if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

string savefilename = saveFileDialog1.FileName;

//File.WriteAllText(savefilename, newfile, Encoding.Default);

using (StreamWriter writetext = new StreamWriter(savefilename, false, Encoding.Default))

{

string[] newfilestrings = newfile.Split('\n');

foreach (string s in newfilestrings)

{

writetext.WriteLine(s);

}

}

/\* неудачная попытка стеганографии в картинки пдф

Bitmap bm = new Bitmap("newimg.png");

byte[ ] data = new byte[3 \* bm.Height \* bm.Width];

for (int i = 0; i < bm.Height; i++)

{

for(int j = 0; j < bm.Width; j++)

{

Color color = bm.GetPixel(j, i);

data[i \* 3 \* bm.Width + j \* 3] = color.R;

data[i \* 3 \* bm.Width + j \* 3 + 1] = color.G;

data[i \* 3 \* bm.Width + j \* 3 + 2] = color.B;

}

}

byte[] newfile = InsertImage(filestring, offsetobj, numobj, data);

File.WriteAllBytes(savefilename, newfile);

\*/

}

else

{

MessageBox.Show("Невозможно скрыть даннный объем информации", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

if (!mode)

{

//функция извлечения

textsource = PdfEncFun(filename, numobj - 1);

textsource = textsource.Substring(0, textsource.Length - 1);

Console.WriteLine(textsource);

if (!textsource.Contains("СТЕГАНОГРАФИЯ"))

{

KeyForm kf = new KeyForm();

kf.f1 = this;

kf.ShowDialog();

pwd = Encoding.Default.GetBytes(key);

//Расшифрование текста сообщения

byte[] bytestextsource = Encoding.Default.GetBytes(textsource);

int sizerand = 8;

Console.WriteLine("зашифрованный поток " + Encoding.Default.GetString(bytestextsource));

enctextsource = Encoding.Default.GetString(DecText(bytestextsource, sizerand));

Console.WriteLine(enctextsource);

if (enctextsource.Contains("ШИФРОВАНИЕ"))

{

textsource = enctextsource.Substring(11);

if (textsource.Contains("КАРТИНКА"))

{

textsource = textsource.Substring(9);

}

}

else

{

MessageBox.Show("Неверный ключ", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

else

{

if (textsource.Contains("КАРТИНКА"))

{

textsource = textsource.Substring(9);

}

}

textsource = textsource.Substring(14);

textsource = textsource.Substring(0, textsource.LastIndexOf(" СТЕГАНОГРАФИЯ"));

if (RBInfText.Checked)

{

TBTypeText.Text = textsource;

}

else if (RBInfFile.Checked)

{

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "txt files (\*.txt)|\*.txt|png files (\*.png)|\*.png";

saveFileDialog1.RestoreDirectory = false;

if (saveFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

string savefilename = saveFileDialog1.FileName;

if (Path.GetExtension(savefilename) == ".png")

{

Console.WriteLine(textsource.Length);

byte[] data = Convert.FromBase64String(textsource);

using (var stream = new MemoryStream(data, 0, data.Length))

{

Image newimage = Image.FromStream(stream);

newimage.Save(savefilename, ImageFormat.Png);

}

}

else if (Path.GetExtension(savefilename) == ".txt")

{

File.WriteAllText(savefilename, textsource, Encoding.Default);

}

}

}

}

}

catch

{

MessageBox.Show("Непредвиденная ошибка", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

AboutForm af;

af = new AboutForm();

af.ShowDialog();

}

private void CBEnc\_CheckedChanged(object sender, EventArgs e)

{

if(CBEnc.Checked)

{

KeyForm kf = new KeyForm();

kf.f1 = this;

kf.ShowDialog();

}

}

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

{

Selectfun sf = new Selectfun();

sf.f1 = this;

sf.ShowDialog();

DES3 = new TripleDESCryptoServiceProvider();

DES3.Padding = PaddingMode.None;

IV = new byte[8] { 0, 0, 0, 0, 0, 0, 0, 0 };// получение случайного начального вектора

DES3.Mode = CipherMode.CFB; // установка режима блочного шифрования

MD4 = new MD4CryptoServiceProvider();

rand = new RNGCryptoServiceProvider();

}

}

}

ImageSteg.cs

using System;

using System.Drawing;

using System.Collections;

namespace КР

{

class ImageSteg

{

private Bitmap image;

private Bitmap newimage;

private int salt;

private string depth;

public ImageSteg(Image img)

{

image = new Bitmap(img);

newimage = new Bitmap(img);

salt = 8 + (((2 \* 3 \* image.Width \* image.Height) / 8) % 8);

depth = img.PixelFormat.ToString();

Console.WriteLine(depth);

Console.WriteLine(salt);

}

public int SaltInf()

{

return salt;

}

public long SizeImage()

{

long size = image.Width \* image.Height;

if (depth == "Format32bppArgb" || depth == "Format24bppRgb")

size = size \* 3; // кол-во пикселей умноженное на кол-во байт под RGB

return size;

}

public Bitmap TextToImage(byte[] bytestext)

{

BitArray bitstext = new BitArray(bytestext);

if(depth == "Format24bppRgb" || depth == "Format32bppArgb")

{

for (int i = 0; i < image.Width; i++)

{

for (int j = 0; j < image.Height; j++)

{

Color pixelColor = image.GetPixel(i, j); //получение цвета пикселя

//значение каждого цвета (0-255)

byte red = pixelColor.R;

byte green = pixelColor.G;

byte blue = pixelColor.B;

//массив бит каждого цвета

BitArray bitsred = new BitArray(8);

BitArray bitsgreen = new BitArray(8);

BitArray bitsblue = new BitArray(8);

//запись битов каждого цвета + скрытие текста в последних 2 битах

for (int k = 0; k < 8; k++)

{

if (k > 1)

{

if ((red >> k & 1) == 1)

bitsred[k] = true;

else

bitsred[k] = false;

if ((green >> k & 1) == 1)

bitsgreen[k] = true;

else

bitsgreen[k] = false;

if ((blue >> k & 1) == 1)

bitsblue[k] = true;

else

bitsblue[k] = false;

}

else

{

int l = k \* 3 + j \* 6 + i \* image.Height \* 6;

if (l < bitstext.Length)

{

bitsred[k] = bitstext[l];

}

else

{

if ((red >> k & 1) == 1)

bitsred[k] = true;

else

bitsred[k] = false;

}

if (l + 1 < bitstext.Length)

{

bitsgreen[k] = bitstext[l + 1];

}

else

{

if ((green >> k & 1) == 1)

bitsgreen[k] = true;

else

bitsgreen[k] = false;

}

if (l + 2 < bitstext.Length)

{

bitsblue[k] = bitstext[l + 2];

}

else

{

if ((blue >> k & 1) == 1)

bitsblue[k] = true;

else

bitsblue[k] = false;

}

}

}

//преобразование массива бит в байты каждого цвета

byte newred = 0;

byte newgreen = 0;

byte newblue = 0;

for (int k = 0; k < 8; k++)

{

if (bitsred[k])

newred += (byte)Math.Pow(2, k);

if (bitsgreen[k])

newgreen += (byte)Math.Pow(2, k);

if (bitsblue[k])

newblue += (byte)Math.Pow(2, k);

}

Color newpixelColor = Color.FromArgb(newred, newgreen, newblue);

newimage.SetPixel(i, j, newpixelColor);

}

}

}

return newimage;

}

public byte[] ImageToText()

{

BitArray bitstext = new BitArray(Convert.ToInt32(SizeImage()\*2));

Console.WriteLine("Распаковка");

byte[] text = new byte[bitstext.Length/8 + 8];

if(depth == "Format24bppRgb" || depth == "Format32bppArgb")

{

for (int i = 0; i < image.Width; i++)

{

for (int j = 0; j < image.Height; j++)

{

Color pixelColor = image.GetPixel(i, j); //получение цвета пикселя

//значение каждого цвета (0-255)

byte red = pixelColor.R;

byte green = pixelColor.G;

byte blue = pixelColor.B;

//массив бит каждого цвета

BitArray bitsred = new BitArray(8);

BitArray bitsgreen = new BitArray(8);

BitArray bitsblue = new BitArray(8);

//запись битов каждого цвета + извлечение текста в последних 2 битах

for (int k = 0; k < 8; k++)

{

if (k > 1)

{

if ((red >> k & 1) == 1)

bitsred[k] = true;

else

bitsred[k] = false;

if ((green >> k & 1) == 1)

bitsgreen[k] = true;

else

bitsgreen[k] = false;

if ((blue >> k & 1) == 1)

bitsblue[k] = true;

else

bitsblue[k] = false;

}

else

{

int l = k \* 3 + j \* 6 + i \* image.Height \* 6;

if ((red >> k & 1) == 1)

bitstext[l] = true;

else

bitstext[l] = false;

if ((green >> k & 1) == 1)

bitstext[l + 1] = true;

else

bitstext[l + 1] = false;

if ((blue >> k & 1) == 1)

bitstext[l + 2] = true;

else

bitstext[l + 2] = false;

}

}

}

}

}

bitstext.CopyTo(text, 0);

Console.WriteLine(text.Length);

return text;

}

}

}

KeyForm.cs

using System;

using System.Drawing;

using System.Windows.Forms;

namespace КР

{

public partial class KeyForm : Form

{

public Form1 f1;

public KeyForm()

{

InitializeComponent();

}

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

{

if (f1.mode)

{

if (TBKey.Text == TBConf.Text)

{

if (TBKey.Text != "")

{

f1.key = TBKey.Text;

Close();

}

else

{

MessageBox.Show("Введен пустой ключ", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

MessageBox.Show("Ключ не совпадает с подтверждением", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

else

{

if (TBKey.Text != "")

{

f1.key = TBKey.Text;

Close();

}

else

{

MessageBox.Show("Введен пустой ключ", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void KeyForm\_Shown(object sender, EventArgs e)

{

if (f1.mode)

{

PanelConf.Visible = true;

PanelKey.Location = new Point(6, 2);

}

else

{

PanelConf.Visible = false;

PanelKey.Location = new Point(6, 20);

}

}

}

}

Selectfun.cs

using System;

using System.Windows.Forms;

namespace КР

{

public partial class Selectfun : Form

{

public Form1 f1;

private bool select = false;

public Selectfun()

{

InitializeComponent();

}

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

{

select = true;

f1.stegimageorpdf = false;

Close();

}

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

{

select = true;

f1.stegimageorpdf = true;

Close();

}

private void Selectfun\_FormClosing(object sender, FormClosingEventArgs e)

{

if(!select)

f1.stegimageorpdf = true;

}

}

}