МИНОБРНАУКИ РОССИИ

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-05-19

Абросимова М. Н.

Москва

2022

Оглавление

Введение

1. Цель курсовой работы

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

2. 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.

3. 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).

Глава 1. 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.

Задача стеганографии в изображениях — встроить информацию в цифровое изображение так, чтобы и сообщение, и сам факт его наличия были скрыты. Полученное изображение с дополнительной скрытой информацией не должно выглядеть аномальным. Это достигается путём внесения изменений, незаметных для человеческого зрения.

1. Алгоритм скрытия информации в графических файлах

LSB (англ. Least Significant Bit — Наименее значимый бит)

Данный метод заключается в выделении наименее значимых бит изображения-контейнера с последующей их заменой на биты сообщения. Поскольку замене подвергаются лишь наименее значимые биты, разница между исходным изображением-контейнером и контейнером, содержащим скрытые данные невелика и обычно незаметна для человеческого глаза. Метод LSB применим лишь к изображениям в форматах без сжатия (например, BMP) или со сжатием без потерь (например, GIF), так как для хранения скрытого сообщения используются наименее значимые биты значений пикселей, при сжатии с потерями эта информация может быть утеряна.

Наиболее популярной цветовой моделью является RGB, где цвет представляется в виде трех составляющих: *красного, зеленого и синего*. Каждая компонента кодируется в классическом варианте с помощью 8 бит, то есть может принимать значение от 0 до 255. Именно здесь и прячется наименее значащий бит. Важно понять, что на один RGB-цвет приходится три бита.

$\{a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0\}$ – отдельный байт, то есть целое число от 0 до 255.

$$A = \sum_{i=0}^{7} a_i \, 2^i$$

a_0 – LSB – наименее значимый бит, его изменение делает $A \pm 1$

a_7 - наиболее значащий бит, его изменение делает число $A \pm 128$

Рассмотрим изображение в формате BMP 24 (Рис 1.1): изображение хранится в трёх матрицах. Это массивы яркостей красного, зеленого и синего цветов (RGB – растровые данные). В массивах хранятся байты, то есть числа 0…255 – значения яркостей цвета. Так (0,0,0) – черный цветы, а (255, 255, 255) – белый цвет.

Рис 1.1 BMP – изображение

Например, фотография размером 100×100 содержит 10000 пикселей. Имеется три матрицы, а значит, фото имеет размер 30000 байт.

Метод LSB использует первое низкоуровневое свойство ЗСЧ (слабая чувствительность к незначительному изменению яркости).

Информационное сообщение встраивается побитно: один бит сообщения в один байт растровых данных посредством записи (замены) LSB.

Если изображение имеет размер 100×100, то можно встроить 30000 информационных бит, потому что 1 пиксель – это 3 байта, но в 1 байте содержится только 1 LSB.

Достоинством метода LSB является:

- высокая пропускная способность ($\frac{1}{8}$ от объема контейнера);
- высокая скорость встраивания и извлечения сообщения и простота реализации стеганосистемы;
- метод LSB может быть расширением до использования двух, трех и более наименее значимых бит. Пропускная способность может быть соответственно $\frac{2}{8}, \frac{3}{8}$. Ведь изменение 3 последних битов изменяет число на ± 7 , а это меньше 3% от 255, то есть для ЗСЧ это практически незаметно.

LSB пустого контейнера распределены, как правило случайно, равновероятно и независимо друг от друга. LSB заполненного контейнера имеют статистику встроенного сообщения. Простейший тест статистики LSB выявит (детектирует) присутствие информационного сообщения.

В реализации курсовой работы будем использовать для скрытия/извлечения информации изображения в формате bmp, не содержащие палитру. В таком bmp файле каждые 3 байта определяют 3 цвета пикселя.

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 в нашем случае).

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

2. Алгоритм шифрования сообщения

Перестановка:

$\forall$ i, $0 \le i \le n-1$ $C_i = P_{k[i]}$, где

- $P=\{P_0, P_1, \dots, P_i, \dots, P_{n-1}\}$ – открытый текст;
- n – длина открытого текста;
- $C=\{C_0, C_1, \dots, C_i, \dots, C_{n-1}\}$ – шифротекст;
- $k=\{k_0, k_1, \dots, k_i, \dots, k_{n-1}\}$ – ключ шифрования.

При расшифровании применяется обратная перестановка:

$\forall$ i, $0 \le i \le n-1$ $P_{k[i]} = C_i$.

При шифровании перестановкой ключ должен удовлетворять условию:

$\forall k_i \in k \; 0 \le k_i \le n-1 \wedge \forall \; k_i, k_j \in k \; k_i \ne k_j$.

Если длина ключа меньше длины открытого текста, то следует разбить открытый текст на блоки, длина которых равна длине ключа, и последовательно применить ключ перестановки к каждому блоку открытого текста. Если длина открытого текста не кратна длине ключа, то последний блок должен быть дополнен пробелами.

Если длина ключа больше длины открытого текста, то ключ усекается до нужной длины.

Процедура генерации ключа перестановки k из текстовой строки (пароля) t:

а) сортировка t по алфавиту и получение новой строки t' (например, t=матрос, t'=аморст);
б) получение элементов ключа k_i=позиция символа t_i в строке t' (например, k=105324);
в) если строка t содержит одинаковые символы (например, t=анна), то при получении элемента ключа k_i используется позиция следующего вхождения символа t_i в строке t' (например, t'=аанн, k=0231) (в программе для этого можно после нахождения позиции очередного символа заменить его, например, символом '\1').

Глава 2. Результаты проектирования

1. Пользовательский интерфейс

Приведем примеры форм оконного приложения, при помощи которого реализовано скрытие и извлечение информации из PDF-файлов (Рис. 2.1.).

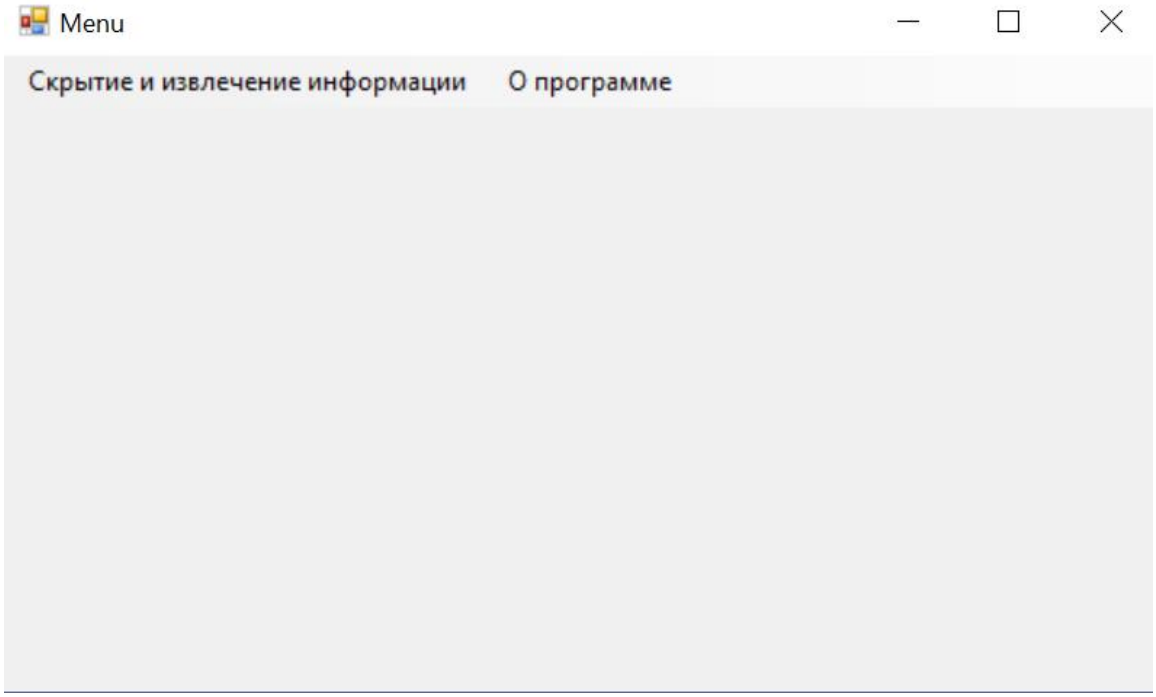


Рис. 2.1. Главная форма оконного приложения

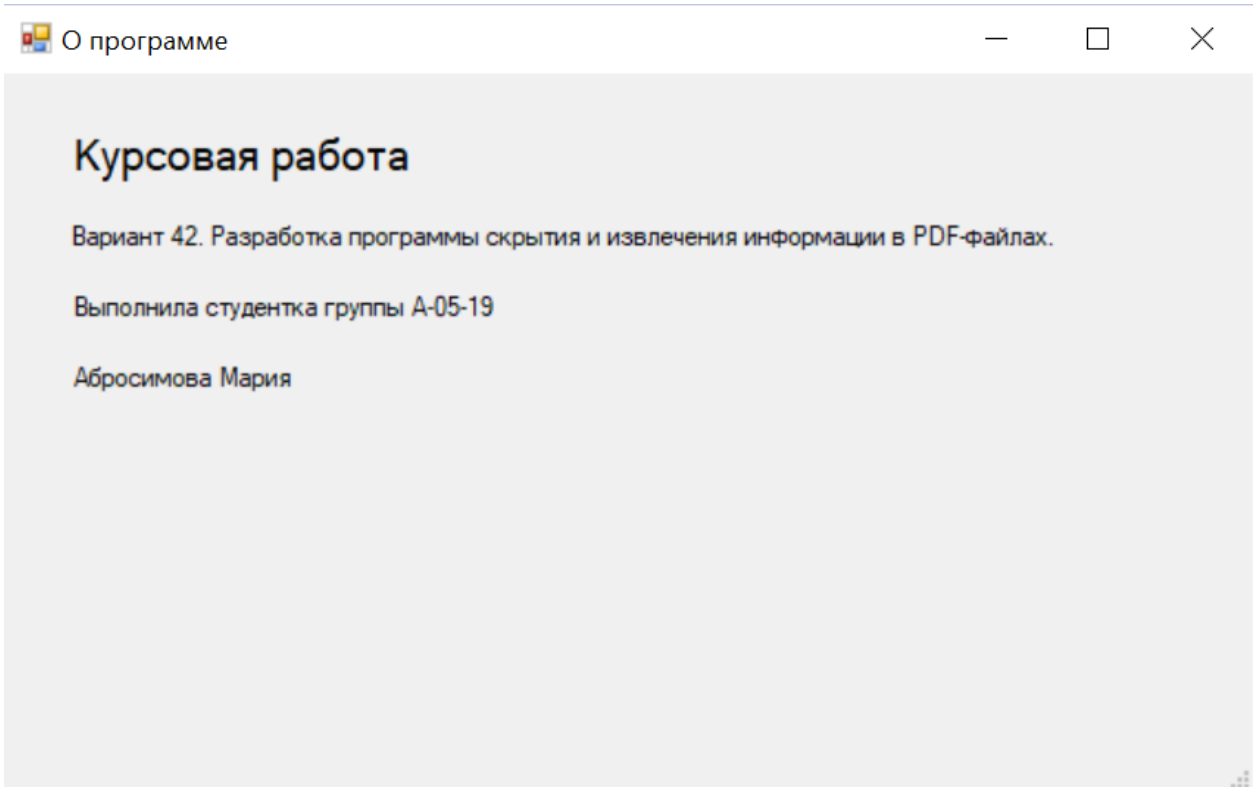


Рис. 2.2. Окно «О программе»

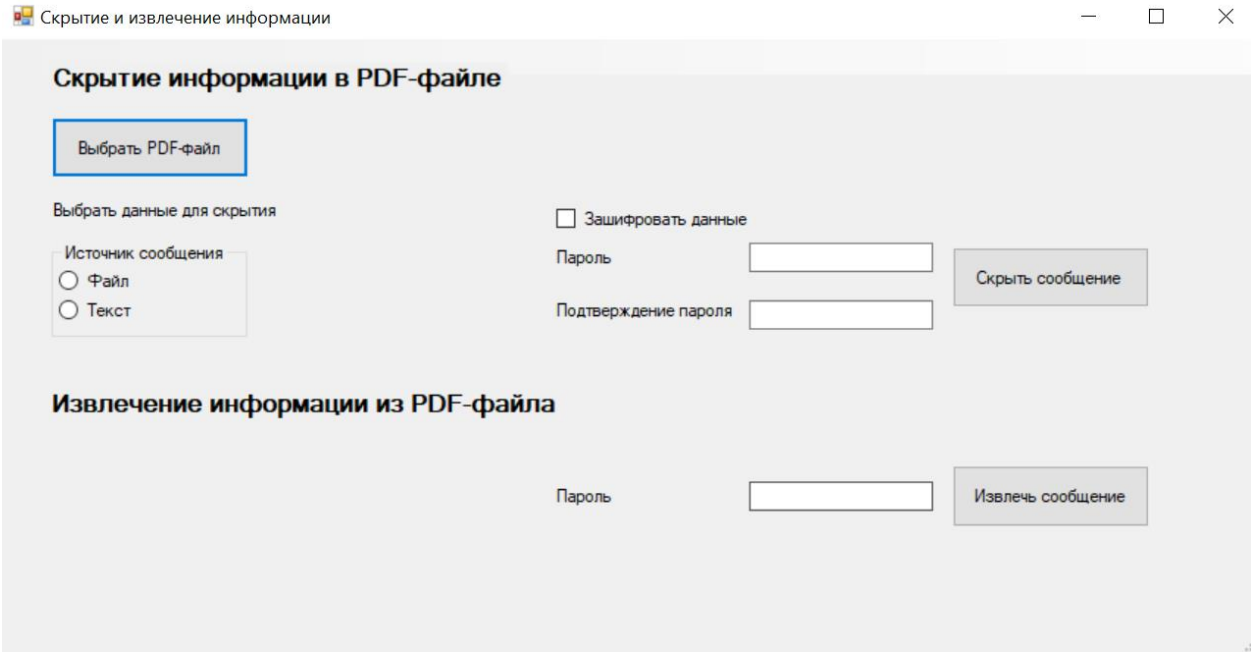


Рис. 2.3. Окно «Скрытие и извлечение информации»

При нажатии кнопки «Выбрать PDF-файл», появляется окно выбора PDF файла-контейнера

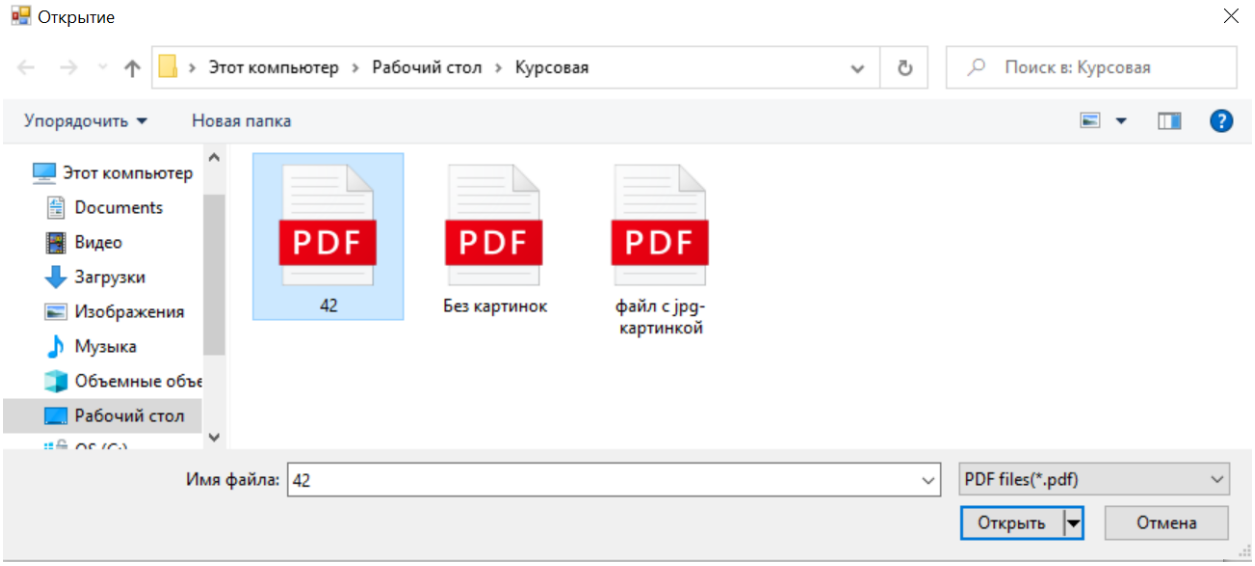


Рис. 2.4. Окно выбора файла-контейнера

При выборе RadioButton «Файл» в GroupBox «Источник сообщения» (Рис. 2.5.), появится окно выбора файла с сообщением (Рис. 2.6.).

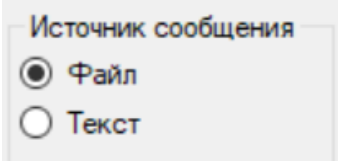


Рис. 2.5. GroupBox выбора источника сообщения

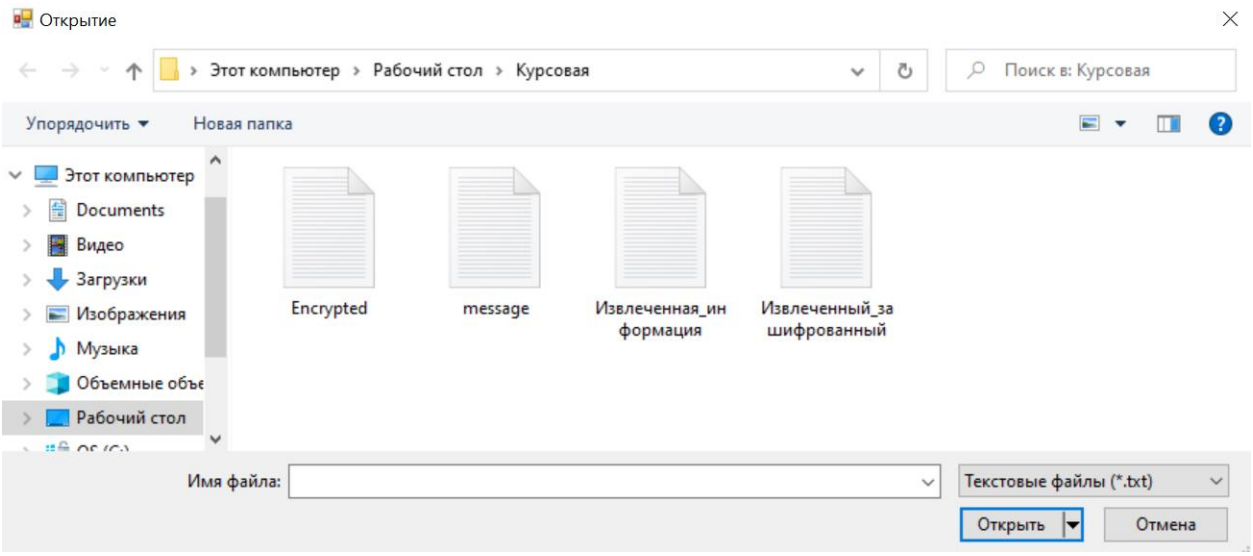


Рис. 2.6. Окно выбора файла с сообщением

При выборе RadioButton «Текст» в GroupBox «Источник сообщения», появится TextBox, в который можно ввести сообщение (Рис. 2.7.).

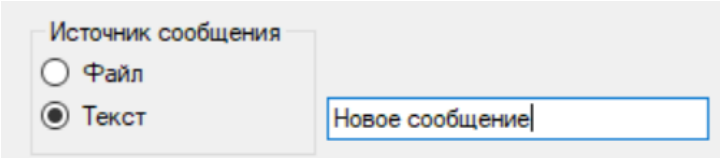


Рис. 2.7. TextBox для ввода сообщения

Нажатие на Checkbox «Зашифровать данные» вызывает появление поля ввода пароля пользователя и подтверждения пароля для шифрования (Рис. 2.8.).

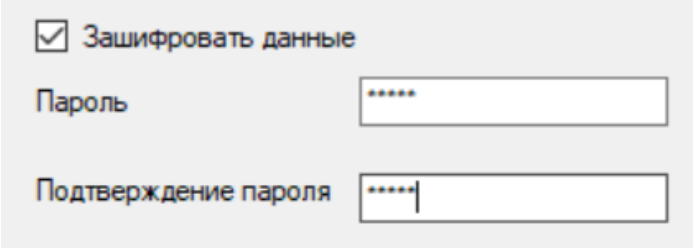


Рис. 2.8. Поля ввода пароля и подтверждения пароля для шифрования

При нажатии на кнопку «Скрыть сообщение» (Рис. 2.9.) появится MessageBox с информацией (Рис. 2.10.).

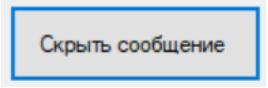


Рис. 2.9. Кнопка «Скрыть сообщение»

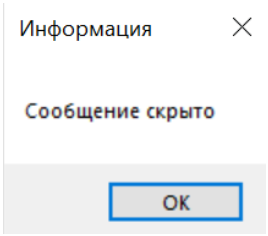


Рис. 2.10. MessageBox «Сообщение скрыто»

При правильном введении пароля и нажатии на кнопку «Извлечь сообщение» (Рис. 2.11.) появится MessageBox с информацией (Рис. 2.12.)

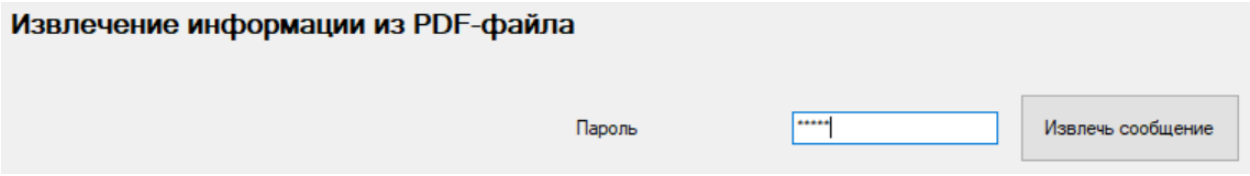


Рис. 2.11. Поле ввода пароля, кнопка «Извлечь сообщение»

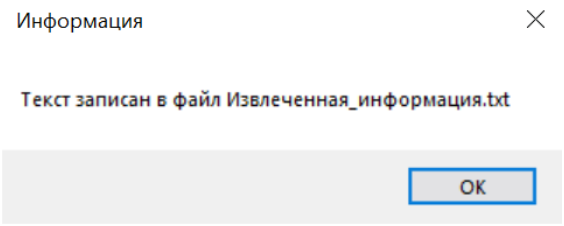


Рис. 2.12. MessageBox с информацией

При возникновении ошибок программа сигнализирует об этом пользователю (Рис. 2.13. , Рис. 2.14. , Рис. 2.15.).

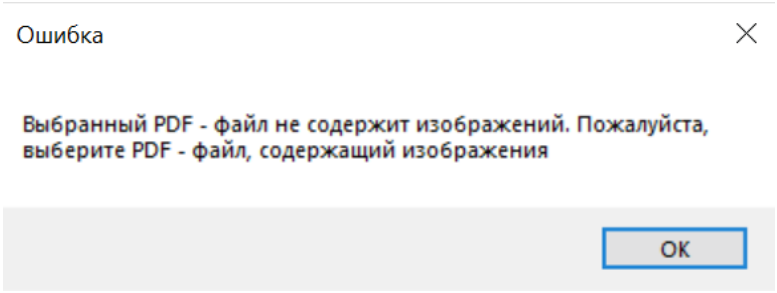


Рис. 2.13. Выбранный PDF-файл не содержит изображений

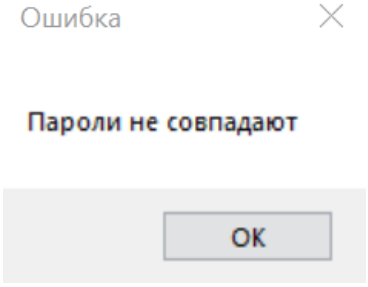


Рис. 2.14. Введенные пароли при скрытии сообщения не совпадают

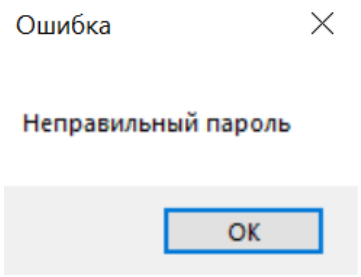


Рис. 2.15. Введен неправильный пароль при попытке извлечь сообщение

2. Программная реализация

2.1 Класс извлечения изображений из PDF-файла

public partial class PDF_ImgExtraction

Поля класса:

- string pdfFile - Название PDF-файла, в котором скрываем сообщение
- PdfReader pdf = new PdfReader(pdfFile);//чтение и анализ pdf-файла программно
- PdfDictionary pg = pdf.GetPageN(pageNumber);// словарь страниц PDF-файла
- PdfDictionary res =(PdfDictionary)PdfReader.GetPdfObject(pg.Get(PdfName.RESOURCES)); -получение ресурсов pdf - файла
- PdfDictionary xobj =(PdfDictionary)PdfReader.GetPdfObject(res.Get(PdfName.XOBJECT)); -изображения в pdf-файлах
- Bitmap d – изображение в формате BMP, извлеченное из PDF-файла
- float width – ширина изображения
- float height – высота изображения
- string imgPath – путь сохранения изображения

Методы класса:

- public void ExtractImage(string pdfFile, string imgPath, bool flag_decode) – извлечение изображения из PDF-файла
- public void RenderImage(ImageRenderInfo renderInfo, int i, string imgPath, bool flag_decode) – сохранение изображения в формате BMP

2.2 Класс Form1

Поля класса:

```
string pdf_filename; //название PDF файла для скрытия
string txt_filename; //название текстового файла с сообщением
string directoryPath; //выбранная директория
string password; //пароль
string key_str = ""; //изначальный пустой пароль
bool is_shifred = false; //флаг проверки шифрования сообщения
string shifr_pdf_filename; //зашифрованный pdf-файл
bool flag_decode = false; //флаг расшифрования
```

Методы класса:

```
//LSB
```

- `private BitArray ByteToBit(byte src)`
- `private byte BitToByte(BitArray scr)`
- `private bool isEncryption(Bitmap scr) - /*Проверяет, зашифрован ли файл, возвраещает true, если символ в первом пикслеле равен / иначе false */`
- `private byte[] NormalizeWriteCount(byte[] CountSymbols) /*Нормализует количество символов для шифрования,чтобы они всегда занимали ENCRYP_TEXT_SIZE байт*/`
- `private void WriteCountText(int count, Bitmap src) - /*Записыает количество символов для шифрования в первые биты картинки */`
- `private int ReadCountText(Bitmap src) - /*Читает количество символов для дешифрования из первых бит картинки*/`

`//шифрование перестановкой на ключе, выводимом из парольной фразы`

- `public string preparing_key_pas(string pas, string pass_key, out string new_pass_key) //подготовка сообщения на основе длины ключа`
- `public int[] Encrypt_perestanovka(string key, out string key_str) //генерация ключа перестановки на основе пароля`
- `public string Code(string pas, int[] key) //шифрование методом перестановки`
- `public string Decode_perestanovka(string pas, string key)//расшифрование`

- `private void Search_pdf_Click(object sender, EventArgs e)`
- `private void RB1_CheckedChanged(object sender, EventArgs e)`
- `private void RB2_CheckedChanged(object sender, EventArgs e)`
- `private void Encrypt_Click(object sender, EventArgs e)`
- `private void Decrypt_Click(object sender, EventArgs e)`
- `private void checkBox1_CheckedChanged(object sender, EventArgs e)`

Глава 3. 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.

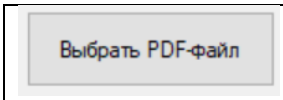


Рис. 3.1. Кнопка выбора PDF-файла

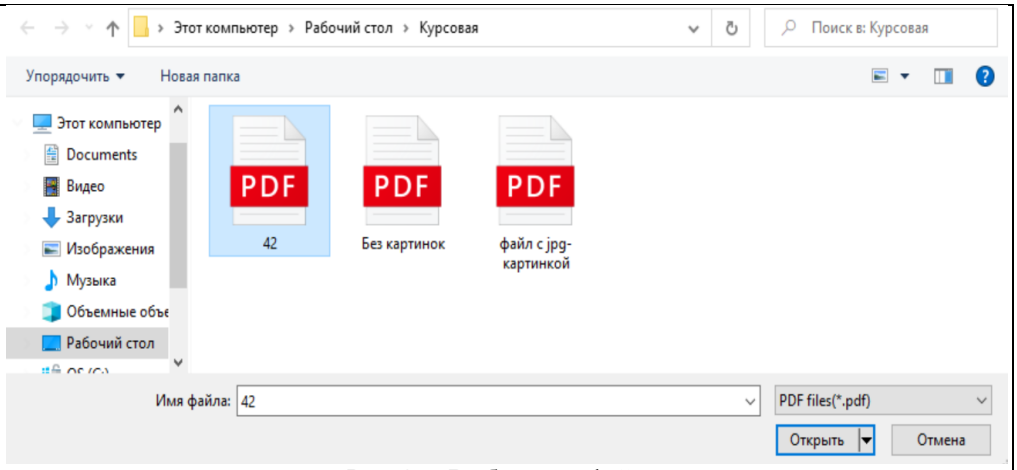
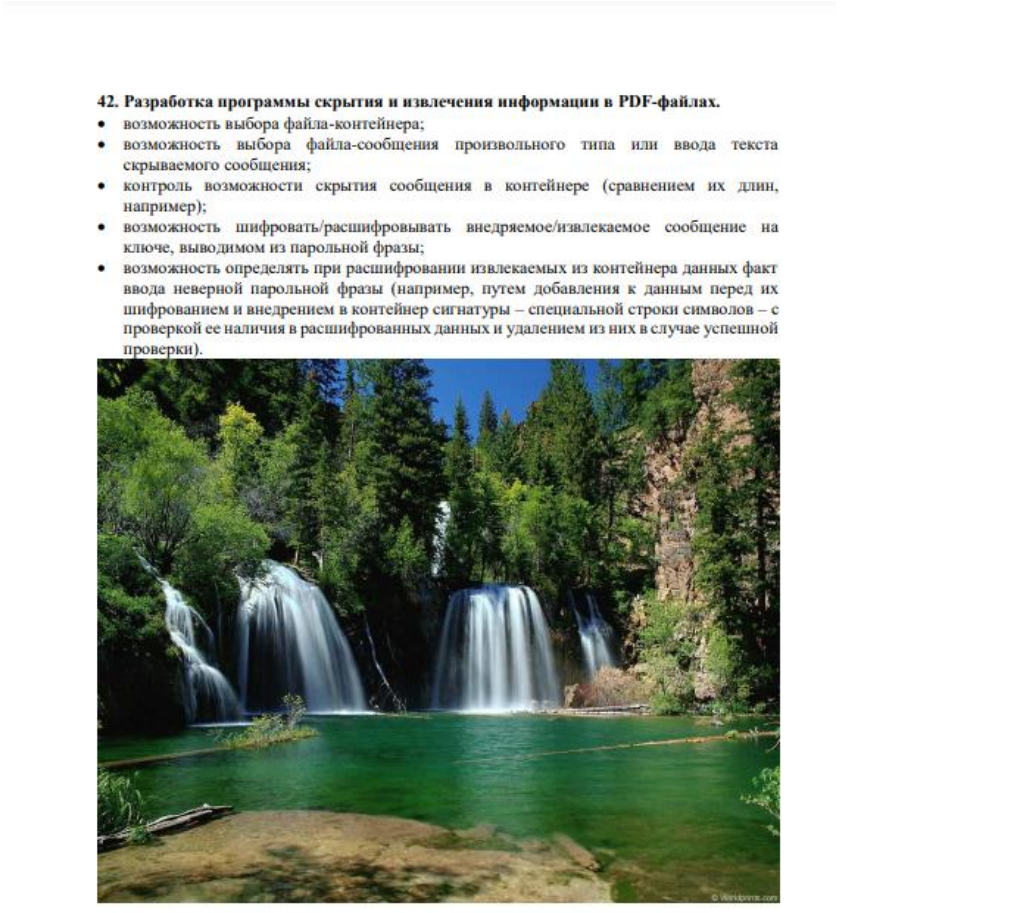


Рис. 3.2. Выбор PDF-файла



42. Разработка программы скрытия и извлечения информации в PDF-файлах.

- 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).

Рис. 3.3. Файл 42.pdf

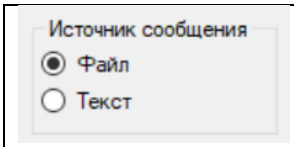


Рис. 3.4. Кнопка выбора файла с сообщением

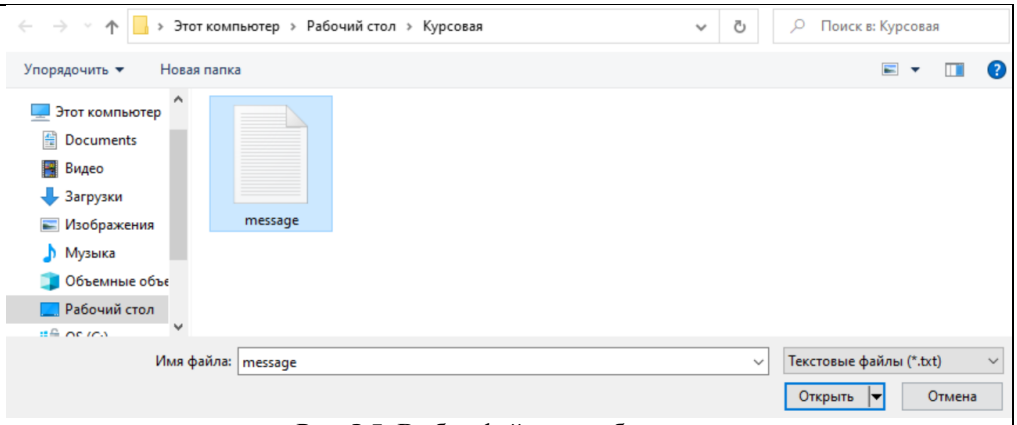


Рис. 3.5. Выбор файла с сообщением

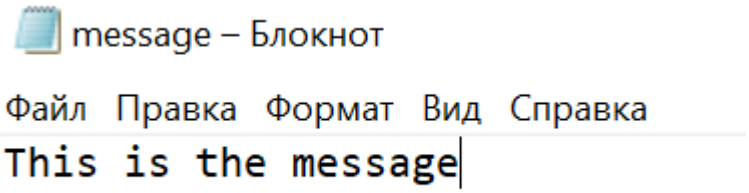


Рис. 3.6. Содержимое файла message.txt

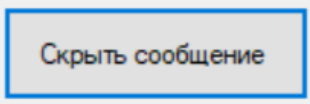


Рис. 3.7. Кнопка скрытия сообщения

42. Разработка программы скрытия и извлечения информации в PDF-файлах.

- 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).

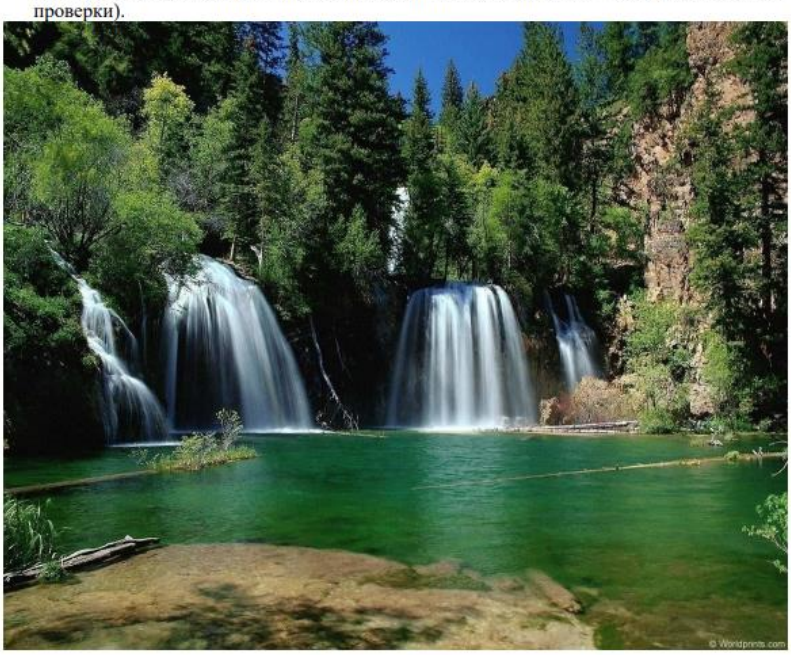


Рис. 3.8. PDF-файл со скрытым изображением

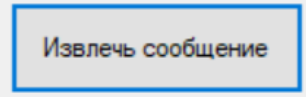


Рис. 3.9. Кнопка скрытия сообщения

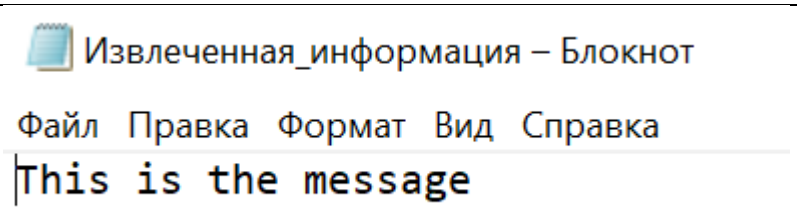


Рис. 3.10. Кнопка скрытия сообщения

Тестовый пример № 2.

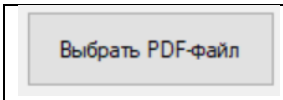


Рис. 3.11. Кнопка выбора PDF-файла

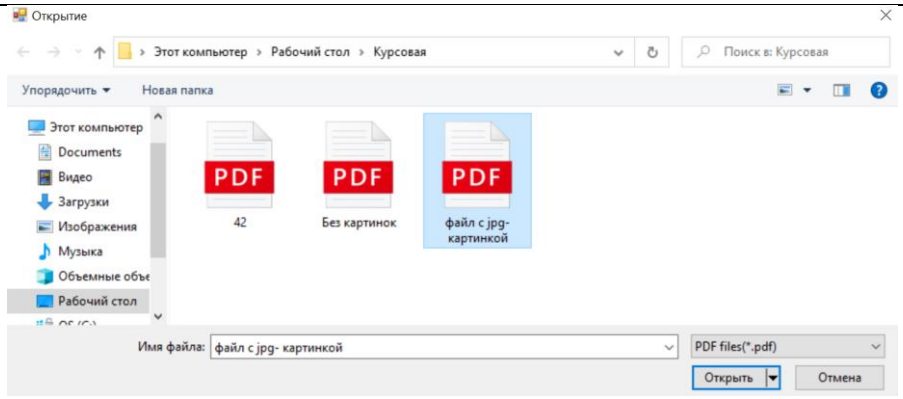


Рис. 3.12. Выбор PDF-файла

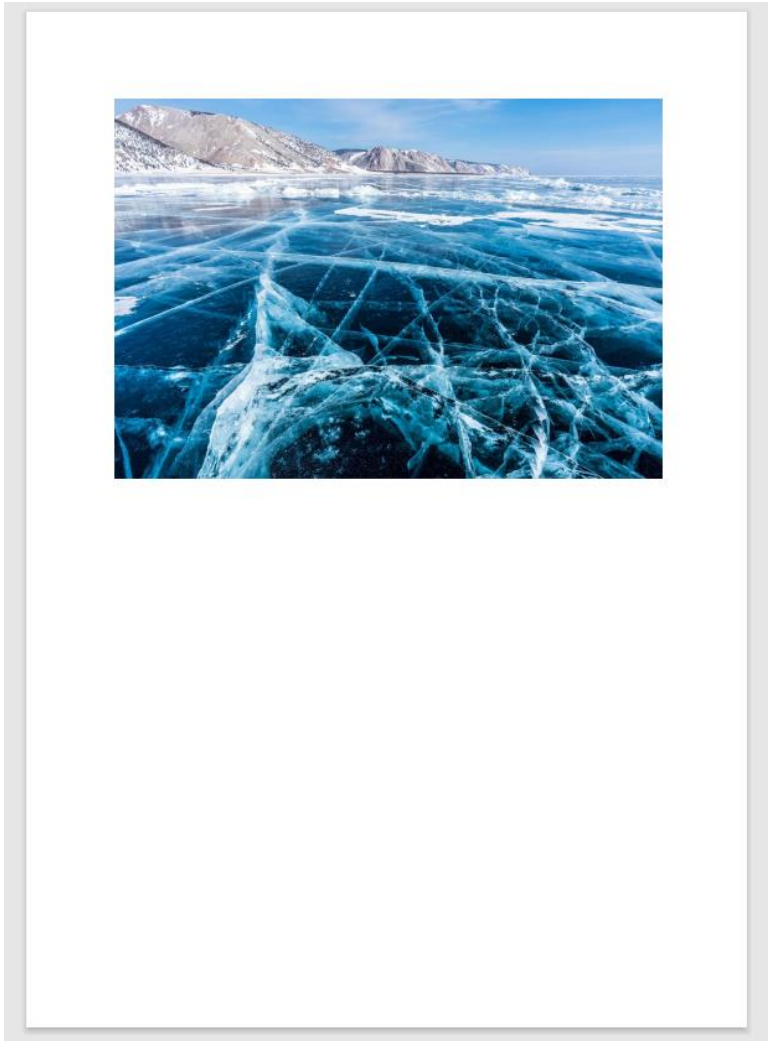


Рис. 3.13. Содержимое выбранного файла

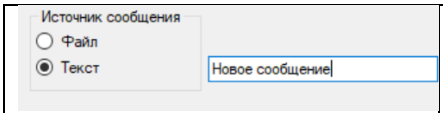


Рис. 3.14. Выбор источника сообщения: текст; ввод сообщения

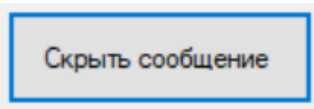


Рис. 3.15. Выбор источника сообщения: текст; ввод сообщения

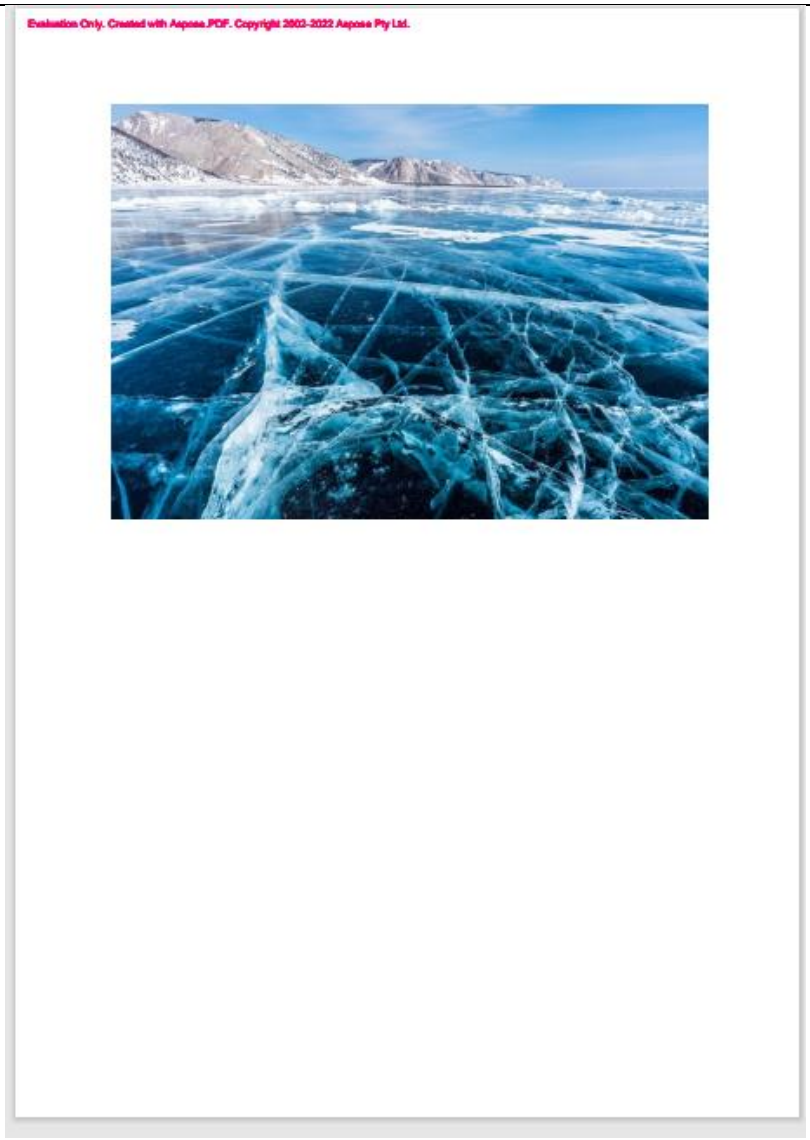


Рис. 3.16. PDF-файл со скрытым изображением

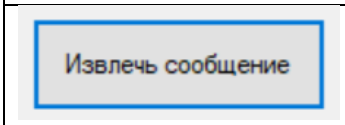


Рис. 3.17. Кнопка извлечения сообщения

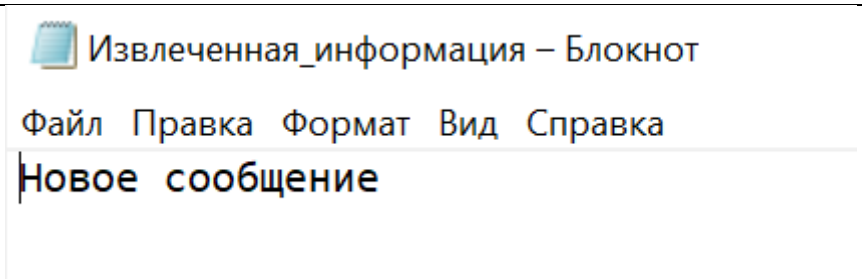


Рис. 3.18. Текстовый файл с извлеченной информацией

Тестовый пример № 3.

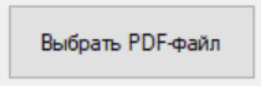


Рис. 3.19. Кнопка выбора PDF-файла

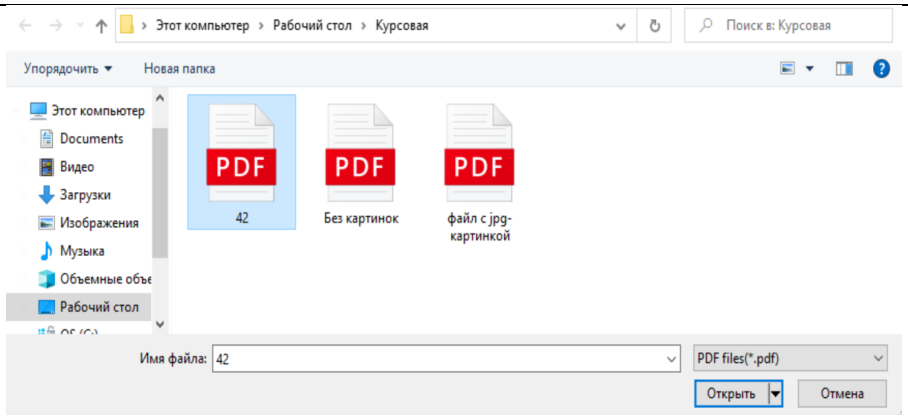


Рис. 3.20. Выбор PDF-файла

42. Разработка программы скрытия и извлечения информации в PDF-файлах.

- 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).

Рис. 3.21. Содержимое PDF-файла 42.pdf

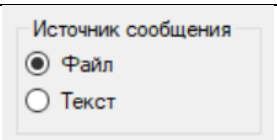


Рис. 3.22. Выбор источника сообщения: файл

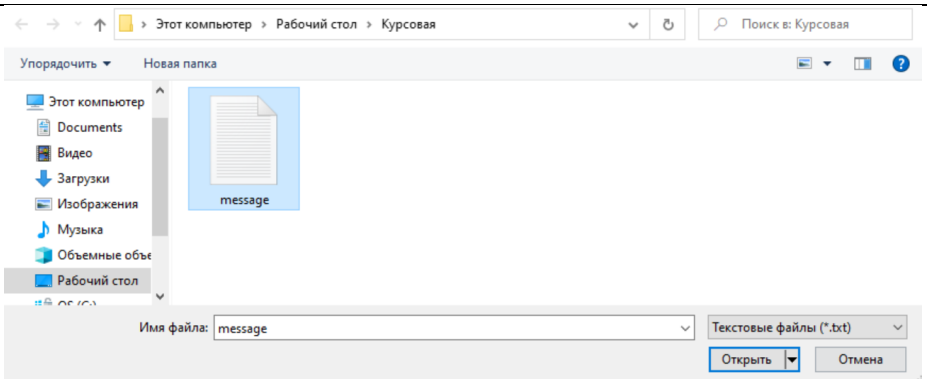


Рис. 3.23. Выбор файла с сообщением

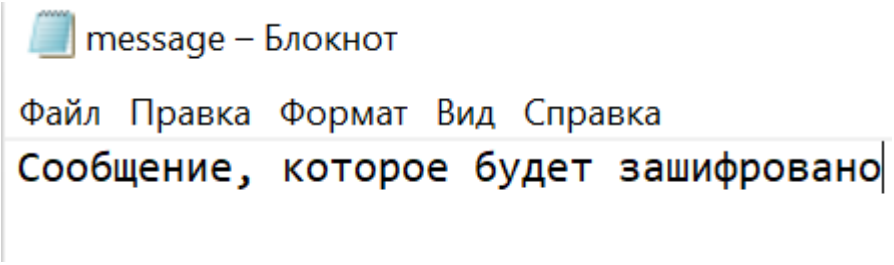


Рис. 3.24. Содержимое файла message.txt

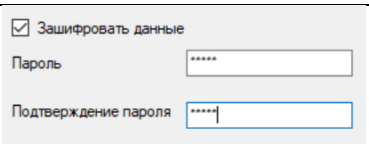


Рис. 3.25. Выбор шифрования сообщения; ввод и подтверждение пароля

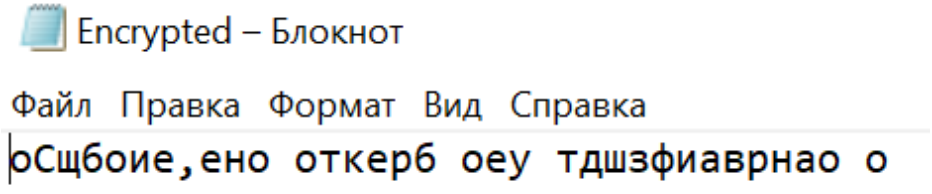


Рис. 3.26. Файл с зашифрованным сообщением

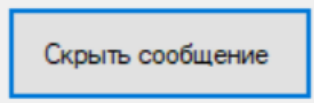


Рис. 3.27. Кнопка скрытия сообщения

42. Разработка программы скрытия и извлечения информации в PDF-файлах.

- 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).

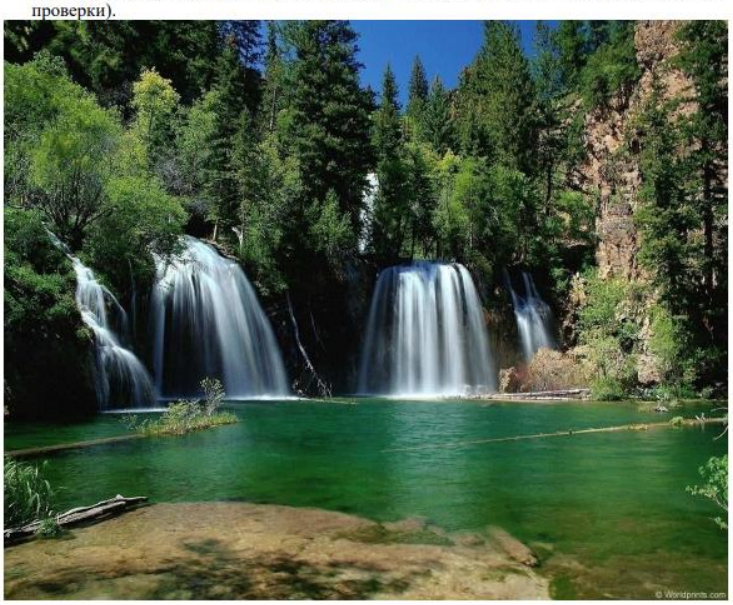


Рис. 3.28. Файл с зашифрованным сообщением

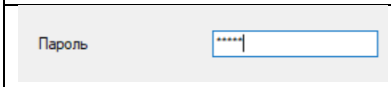


Рис. 3.29. Поле ввода пароля

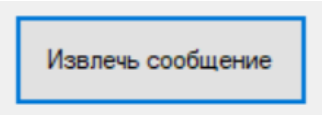


Рис. 3.30. Кнопка извлечения сообщения

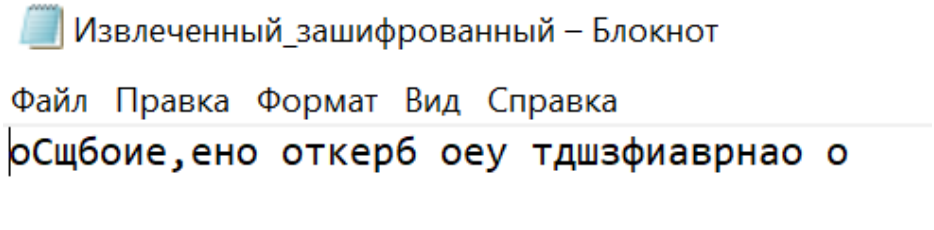


Рис. 3.31.Текстовый файл с извлеченным зашифрованным сообщением

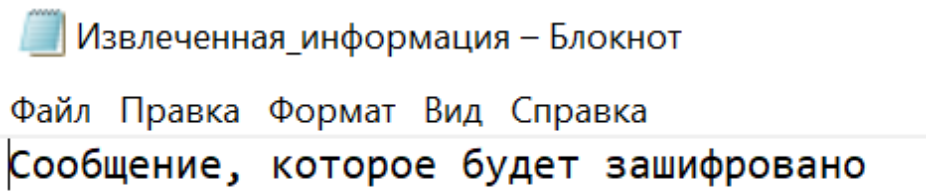


Рис. 3.32.Текстовый файл с извлеченным расшифрованным сообщением

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.
- Произведено ознакомление с библиотеками работы с PDF-файлами в C#: iTextSharp.text.pdf; BitMiracle.Docotic.Pdf; Aspose.Pdf;
- 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. Стеганография. Скрываем текстовую информацию в bmp файле. Практическая реализация на C#

   URL: https://habr.com/ru/post/140373/ (дата обращения - 30.11.2022)

2. Обзор C# библиотек для работы с PDF

   URL: https://habr.com/ru/post/112707/ (дата обращения - 30.11.2022)

3. LSB-стеганография

   URL: https://habr.com/ru/post/422593/ (дата обращения - 30.11.2022)

4. iTextSharp

   URL: https://billibook.blogspot.com/2012/08/iTextSharp.html (дата обращения - 30.11.2022)

5. Extract image from PDF using itextsharp

   URL: https://stackoverflow.com/questions/5945244/extract-image-from-pdf-using-itextsharp (дата обращения - 30.11.2022)

6. Добавление, удаление, извлечение и замена изображений в PDF с помощью C#

   URL: https://blog.aspose.com/ru/pdf/work-with-images-in-pdf-in-csharp/ (дата обращения - 30.11.2022)

Приложение. Листинг программы

Form1.cs

```
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;

using iTextSharp.text;
using iTextSharp.text.pdf;
using iTextSharp.text.pdf.parser;
using System.Drawing.Imaging;
using System.IO;
using iTextSharp.text.pdf.parser;
using Dotnet = System.Drawing.Image;
using iTextSharp.text.pdf;
using iTextSharp.text;
using iTextSharp.text.pdf;
using Aspose.Pdf;
using System.Collections;
using BitMiracle.Docotic.Pdf;
using System.IO;
namespace CURS_PDF_Encoder

{
    public partial class Form1 : Form
    {
        string pdf_filename; //название PDF файла для скрытия
        string txt_filename; //название текстового файла с сообщением
        string directoryPath; //выбранная директория
        string password; //пароль
        string key_str = ""; //изначальный пустой пароль
        bool is_shifred = false; //флаг проверки шифрования сообщения
        string shifr_pdf_filename; //зашифрованный pdf-файл
        bool flag_decode = false; //флаг расшифрования
        string directoryPath_shifr_pdf;
        public Form1()
        {
            InitializeComponent();
        }

        //-----------------------------------------------------методы-----------------------------------
--------
        //класс извлечения из pdf-файла bitmap-ов с их последующим сохранением
        public partial class PDF_ImgExtraction
        {
            //string imgPath = "C://Users//Мария//Desktop//Курсовая//";
            //извлекаем из PDF файла изображения
            public void ExtractImage(string pdfFile, string imgPath, bool flag_decode)
            {
                int i = 0;
                PdfReader pdfReader = new PdfReader(pdfFile);
                for (int pageNumber = 1; pageNumber <= pdfReader.NumberOfPages; pageNumber++)
                {
                    PdfReader pdf = new PdfReader(pdfFile);
                    PdfDictionary pg = pdf.GetPageN(pageNumber);
                    PdfDictionary res =
(PdfDictionary)PdfReader.GetPdfObject(pg.Get(PdfName.RESOURCES));
                    PdfDictionary xobj =
(PdfDictionary)PdfReader.GetPdfObject(res.Get(PdfName.XOBJECT));

                    if (xobj != null)
                    {
                        foreach (PdfName name in xobj.Keys)
                        {
                            i++;
                            PdfObject obj = xobj.Get(name);
                            if (obj.IsIndirect())
                            {
                                PdfDictionary tg = (PdfDictionary)PdfReader.GetPdfObject(obj);
```

```
                                string width = tg.Get(PdfName.WIDTH).ToString();
                                string height = tg.Get(PdfName.HEIGHT).ToString();
                                ImageRenderInfo imgRI = ImageRenderInfo.CreateForXObject(new
iTextSharp.text.pdf.parser.Matrix(float.Parse(width), float.Parse(height)), (PRIndirectReference)obj,
tg);
                                RenderImage(imgRI, i, imgPath, flag_decode);
                                PdfReader.KillIndirect(obj);


                            }
                        }
                    }
                    else { MessageBox.Show("Выбранный PDF - файл не содержит изображений. Пожалуйста,
выберите PDF - файл, содержащий изображения", "Ошибка"); }
                }
            }
            //сохраняем изображения, полученные в ExtractImage в формате bmp
            public void RenderImage(ImageRenderInfo renderInfo, int i, string imgPath, bool flag_decode)
            {
                PdfImageObject image = renderInfo.GetImage();
                using (Dotnet dotnetImg = image.GetDrawingImage())
                {
                    if (dotnetImg != null)
                    {
                        using (MemoryStream ms = new MemoryStream())
                        {
                            dotnetImg.Save(ms, ImageFormat.Tiff);
                            Bitmap d = new Bitmap(dotnetImg);
                            if (flag_decode)
                            { d.Save(imgPath + "\\" + "decode" + i + ".bmp"); } //,
System.Drawing.Imaging.ImageFormat.Bmp
                            else { d.Save(imgPath + "\\" + i + ".bmp",
System.Drawing.Imaging.ImageFormat.Bmp); }
                        }
                    }
                }
            }



        }



        //-----------------------------------------------------Least Significant Bit---------------------
-----------------------
        const int ENCRYP_PESENT_SIZE = 1;//текущая
        const int ENCRYP_TEXT_SIZE = 3;
        const int ENCRYP_TEXT_MAX_SIZE = 999;

        private BitArray ByteToBit(byte src)
        {
            BitArray bitArray = new BitArray(8);
            bool st = false;
            for (int i = 0; i < 8; i++)
            {
                if ((src >> i & 1) == 1)
                {
                    st = true;
                }
                else st = false;
                bitArray[i] = st;
            }
            return bitArray;
        }

        private byte BitToByte(BitArray scr)
        {
            byte num = 0;
            for (int i = 0; i < scr.Count; i++)
                if (scr[i] == true)
                    num += (byte)Math.Pow(2, i);
            return num;
        }

        /*Проверяет, зашифрован ли файл,  возвращает true, если символ в первом пикслеле равен / иначе
```

```
false */
        private bool isEncryption(Bitmap scr)
        {
            byte[] rez = new byte[1];
            System.Drawing.Color color = scr.GetPixel(0, 0);
            BitArray colorArray = ByteToBit(color.R); //получаем байт цвета и преобразуем в массив бит
            BitArray messageArray = ByteToBit(color.R); ;//инициализируем результирующий массив бит
            messageArray[0] = colorArray[0];
            messageArray[1] = colorArray[1];

            colorArray = ByteToBit(color.G);//получаем байт цвета и преобразуем в массив бит
            messageArray[2] = colorArray[0];
            messageArray[3] = colorArray[1];
            messageArray[4] = colorArray[2];

            colorArray = ByteToBit(color.B);//получаем байт цвета и преобразуем в массив бит
            messageArray[5] = colorArray[0];
            messageArray[6] = colorArray[1];
            messageArray[7] = colorArray[2];
            rez[0] = BitToByte(messageArray); //получаем байт символа, записанного в 1 пикселе
            string m = Encoding.GetEncoding(1251).GetString(rez);
            if (m == "/")
            {
                return true;
            }
            else return false;
        }

        /*Нормализует количество символов для шифрования,чтобы они всегда занимали ENCRYP_TEXT_SIZE
байт*/
        private byte[] NormalizeWriteCount(byte[] CountSymbols)
        {
            int PaddingByte = ENCRYP_TEXT_SIZE - CountSymbols.Length;

            byte[] WriteCount = new byte[ENCRYP_TEXT_SIZE];

            for (int j = 0; j < PaddingByte; j++)
            {
                WriteCount[j] = 0x30;
            }

            for (int j = PaddingByte; j < ENCRYP_TEXT_SIZE; j++)
            {
                WriteCount[j] = CountSymbols[j - PaddingByte];
            }
            return WriteCount;
        }

        /*Записыает количество символов для шифрования в первые биты картинки */
        private void WriteCountText(int count, Bitmap src)
        {
            byte[] CountSymbols = Encoding.GetEncoding(1251).GetBytes(count.ToString());

            if (CountSymbols.Length < ENCRYP_TEXT_SIZE)
            {
                CountSymbols = NormalizeWriteCount(CountSymbols);
            }

            for (int i = 0; i < ENCRYP_TEXT_SIZE; i++)
            {
                BitArray bitCount = ByteToBit(CountSymbols[i]); //биты количества символов
                System.Drawing.Color pColor = src.GetPixel(0, i + 1);
                BitArray bitsCurColor = ByteToBit(pColor.R); //бит цветов текущего пикселя
                bitsCurColor[0] = bitCount[0];
                bitsCurColor[1] = bitCount[1];
                byte nR = BitToByte(bitsCurColor); //новый бит цвета пиксея

                bitsCurColor = ByteToBit(pColor.G);//бит бит цветов текущего пикселя
                bitsCurColor[0] = bitCount[2];
                bitsCurColor[1] = bitCount[3];
                bitsCurColor[2] = bitCount[4];
                byte nG = BitToByte(bitsCurColor);//новый цвет пиксея

                bitsCurColor = ByteToBit(pColor.B);//бит бит цветов текущего пикселя
                bitsCurColor[0] = bitCount[5];
                bitsCurColor[1] = bitCount[6];
```

```
                bitsCurColor[2] = bitCount[7];
                byte nB = BitToByte(bitsCurColor);//новый цвет пиксея

                System.Drawing.Color nColor = System.Drawing.Color.FromArgb(nR, nG, nB); //новый цвет из
полученных битов
                src.SetPixel(0, i + 1, nColor); //записали полученный цвет в картинку
            }
        }

        /*Читает количество символов для дешифрования из первых бит картинки*/
        private int ReadCountText(Bitmap src)
        {
            byte[] rez = new byte[ENCRYP_TEXT_SIZE];
            for (int i = 0; i < ENCRYP_TEXT_SIZE; i++)
            {
                System.Drawing.Color color = src.GetPixel(0, i + 1);
                BitArray colorArray = ByteToBit(color.R); //биты цвета
                BitArray bitCount = ByteToBit(color.R); ; //инициализация результирующего массива бит
                bitCount[0] = colorArray[0];
                bitCount[1] = colorArray[1];

                colorArray = ByteToBit(color.G);
                bitCount[2] = colorArray[0];
                bitCount[3] = colorArray[1];
                bitCount[4] = colorArray[2];

                colorArray = ByteToBit(color.B);
                bitCount[5] = colorArray[0];
                bitCount[6] = colorArray[1];
                bitCount[7] = colorArray[2];
                rez[i] = BitToByte(bitCount);
            }
            string m = Encoding.GetEncoding(1251).GetString(rez);
            return Convert.ToInt32(m, 10);
        }
        //---------------------------------шифрование перестановкой на ключе, выводимом из парольной
фразы;----------------------------
        public string preparing_key_pas(string pas, string pass_key, out string new_pass_key)
        { //подготовка сообщения на основе длины ключа
            string encrypted_password = string.Empty;
            int key_length = pass_key.Length;
            int pas_length = pas.Length;
            string new_pas = pas.Substring(0);

            if (key_length > pas_length)
            {
                new_pass_key = pass_key.Substring(0, pas_length); //Если длина ключа больше длины
открытого текста, то ключ усекается до нужной длины.

            }
            else
            {
                new_pass_key = pass_key;
                int x = pas_length % key_length;
                if (x != 0)
                {
                    int dop = key_length - x; //сколько нулей нужно добавить в последний блок
                    new_pas += new string(' ', dop);
                }


            }
            return new_pas;
        }
        //генерация ключа перестановки на основе пароля
        public int[] Encrypt_perestanovka(string key, out string key_str)
        {
            string sort_alp = String.Concat(key.OrderBy(ch => ch)); //отсортированный ключ  //сортировка
пароля по алфавиту

            int[] position = new int[key.Length];
            for (int i = 0; i < key.Length; i++)
            {
                int m = sort_alp.IndexOf(key[i]);
                position[i] = m;    //позиция символа старой строки в строке новой
                StringBuilder sb = new StringBuilder(sort_alp);
```

```
                sb[m] = '@';
                sort_alp = sb.ToString();
            }
            key_str = string.Join("", position); //пробел
            return position;
        }

        //шифрование методом перестановки
        public string Code(string pas, int[] key)
        {
            int n = pas.Length / key.Length;
            string buf_block = "";
            string result = "";
            int begin = 0;

            for (int i = 0; i < n; i++)
            {
                buf_block = pas.Substring(begin, key.Length);
                for (int j = 0; j < key.Length; j++)
                    result += buf_block[key[j]];
                begin += key.Length;
            }

            return result;
        }

        //расшифрование
        public string Decode_perestanovka(string pas, string key)
        {
            string decoded_pas = "";

            int n = pas.Length / key.Length;
            string buf_block = "";
            string result = "";
            string res = "";
            int a;
            a = 0;
            char buf;
            for (int i = 0; i < n; i++)
            {
                Dictionary<int, string> elem = new Dictionary<int, string>();
                buf_block = pas.Substring(a, key.Length);
                for (int j = 0; j < key.Length; j++)
                {   //Преобразует строковое представление числа в эквивалентное ему 32-битовое целое
число со знаком
                    elem.Add(Int32.Parse(char.ToString(key[j])), buf_block.Substring(j, 1));
                }
                foreach (var item in elem.OrderBy(x => x.Key))
                {
                    result += item.Value;
                }

                a += key.Length;



            }
            //result = result.Replace(" ", ""); //не удаляем пробелы

            return result;
        }

        //-----------------------------------------------------реализация-------------------------------
------------
        private void Search_pdf_Click(object sender, EventArgs e)
        {  //выбор pdf-файла
            flag_decode = false;
            PDF_ImgExtraction PDF1 = new PDF_ImgExtraction();

            OpenFileDialog OPF = new OpenFileDialog();
            OPF.Filter = "PDF files(*.pdf)|*.pdf";

            if (OPF.ShowDialog() == DialogResult.Cancel)
                return;
            pdf_filename = OPF.FileName;
            directoryPath = System.IO.Path.GetDirectoryName(pdf_filename);//файл содержится в папке
```

```
            //MessageBox.Show(directoryPath);
            //извлечение из pdf-файла картинок и сохранение из в формате bmp
            PDF1.ExtractImage(pdf_filename, directoryPath, flag_decode);
        }

        private void RB1_CheckedChanged(object sender, EventArgs e)
        {
            if (this.RB1.Checked)
            {
                OpenFileDialog dText = new OpenFileDialog();
                dText.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
                if (dText.ShowDialog() == DialogResult.OK)
                {
                    txt_filename = dText.FileName;
                    // MessageBox.Show(txt_filename);
                }
                else
                {
                    txt_filename = "";
                    return;
                }


            }
        }

        private void RB2_CheckedChanged(object sender, EventArgs e)
        {
            if (this.RB2.Checked)
            {
                TB1.Visible = true;
            }
            else TB1.Visible = false;
        }


        private void Encrypt_Click(object sender, EventArgs e)
        {
            //string message = "";
            string preparing_message = "";
            if (RB2.Checked)
            {
                TB1.Visible = true;
                FileStream file3 = new FileStream(directoryPath + "//" + "input_message.txt",
FileMode.Create);
                StreamWriter fnew3 = new StreamWriter(file3);
                fnew3.WriteLine(TB1.Text);
                fnew3.Close();
                file3.Close();
                txt_filename = directoryPath + "//" + "input_message.txt";
            }


            if ((checkBox1.Checked) &&  (Pas1.Text.Length == 0)) { MessageBox.Show("Введите пароль",
"Ошибка"); }
            else if ((checkBox1.Checked) && (Pas1.Text != Pas2.Text))  { MessageBox.Show("Пароли не
совпадают", "Ошибка"); }

            else
            { //try выбрать файл нужно обязательно

                if ((checkBox1.Checked) && (Pas1.Text.Length != 0) && (Pas1.Text == Pas2.Text))
                {
                    MessageBox.Show("Шифруем сообщение");
                    is_shifred = true;
                    StreamReader f = new StreamReader(txt_filename);
                    FileStream file2 = new FileStream(directoryPath + "//" + "Encrypted.txt",
FileMode.Create);
                    StreamWriter fnew2 = new StreamWriter(file2);
                    while (!f.EndOfStream)
                    {
                        string message = f.ReadLine();
                        password = Pas1.Text;

                        preparing_message = preparing_key_pas(message/*сообщение*/, password  /*ключ
шифррования*/, out preparing_message);//+
```

```
                        int[] position;
                        position = Encrypt_perestanovka(password, out key_str);//
                        string key_sort = string.Join("", preparing_message);//key;
                        string encrypt_pass = Code(preparing_message, position);
                        //fnew2.WriteLine(encrypt_pass + "#preparing_message" + preparing_message +
"#key_str " + key_str + "#pas" + password);
                        fnew2.WriteLine(encrypt_pass);
                    }
                    f.Close();
                    fnew2.Close();
                }
                string FilePic = directoryPath + "//1.bmp";
                string FileText = "";
                if (is_shifred) { FileText = directoryPath + "//" + "Encrypted.txt"; }
                else FileText = txt_filename; //directoryPath + "//" + "Encrypted.txt";//txt_filename;
//зашифрованный файл скрываем в bmp
                FileStream rFile;
                try
                {
                    rFile = new FileStream(FilePic, FileMode.Open); //открываем поток
                }
                catch (IOException)
                {
                    MessageBox.Show("Ошибка открытия изображения", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    return;
                }
                Bitmap bPic = new Bitmap(rFile);

                FileStream rText;
                try
                {
                    rText = new FileStream(FileText, FileMode.Open); //открываем поток
                }
                catch (IOException)
                {
                    MessageBox.Show("Ошибка открытия текстового файла", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    return;
                }



                BinaryReader bText = new BinaryReader(rText, Encoding.ASCII);

                List<byte> bList = new List<byte>();
                while (bText.PeekChar() != -1)
                { //считали весь текстовый файл для шифрования в лист байт
                    bList.Add(bText.ReadByte());
                }
                int CountText = bList.Count; // в CountText - количество в байтах текста, который нужно
закодировать
                bText.Close();
                rFile.Close();

                //проверям, что размер не выходит за рамки максимального, поскольку для хранения размера
используется
                //ограниченное количество байт
                if (CountText > (ENCRYP_TEXT_MAX_SIZE - ENCRYP_PESENT_SIZE - ENCRYP_TEXT_SIZE))
                {
                    MessageBox.Show("Размер текста велик для данного алгоритма, уменьшите размер",
"Информация", MessageBoxButtons.OK);
                    return;
                }

                //проверяем, поместится ли исходный текст в картинке
                if (CountText > (bPic.Width * bPic.Height))
                {
                    MessageBox.Show("Выбранная картинка мала для размещения выбранного текста",
"Информация", MessageBoxButtons.OK);
                    return;
                }

                //проверяем, может быть картинка уже зашифрована
                if (isEncryption(bPic))
```

```
                {
                    MessageBox.Show("Файл уже зашифрован", "Информация", MessageBoxButtons.OK);
                    return;
                }

                byte[] Symbol = Encoding.GetEncoding(1251).GetBytes("/");
                BitArray ArrBeginSymbol = ByteToBit(Symbol[0]);
                System.Drawing.Color curColor = bPic.GetPixel(0, 0);
                BitArray tempArray = ByteToBit(curColor.R);
                tempArray[0] = ArrBeginSymbol[0];
                tempArray[1] = ArrBeginSymbol[1];
                byte nR = BitToByte(tempArray);

                tempArray = ByteToBit(curColor.G);
                tempArray[0] = ArrBeginSymbol[2];
                tempArray[1] = ArrBeginSymbol[3];
                tempArray[2] = ArrBeginSymbol[4];
                byte nG = BitToByte(tempArray);

                tempArray = ByteToBit(curColor.B);
                tempArray[0] = ArrBeginSymbol[5];
                tempArray[1] = ArrBeginSymbol[6];
                tempArray[2] = ArrBeginSymbol[7];
                byte nB = BitToByte(tempArray);

                System.Drawing.Color nColor = System.Drawing.Color.FromArgb(nR, nG, nB);
                bPic.SetPixel(0, 0, nColor);
                //то есть в первом пикселе будет символ /, который говорит о том, что картинка
зашифрована

                WriteCountText(CountText, bPic); //записываем количество символов для шифрования

                int index = 0;
                bool st = false;
                for (int i = ENCRYP_TEXT_SIZE + 1; i < bPic.Width; i++)
                {
                    for (int j = 0; j < bPic.Height; j++)
                    {
                        System.Drawing.Color pixelColor = bPic.GetPixel(i, j);
                        if (index == bList.Count)
                        {
                            st = true;
                            break;
                        }
                        BitArray colorArray = ByteToBit(pixelColor.R);
                        BitArray messageArray = ByteToBit(bList[index]);
                        colorArray[0] = messageArray[0]; //меняем
                        colorArray[1] = messageArray[1]; // в нашем цвете биты
                        byte newR = BitToByte(colorArray);

                        colorArray = ByteToBit(pixelColor.G);
                        colorArray[0] = messageArray[2];
                        colorArray[1] = messageArray[3];
                        colorArray[2] = messageArray[4];
                        byte newG = BitToByte(colorArray);

                        colorArray = ByteToBit(pixelColor.B);
                        colorArray[0] = messageArray[5];
                        colorArray[1] = messageArray[6];
                        colorArray[2] = messageArray[7];
                        byte newB = BitToByte(colorArray);

                        System.Drawing.Color newColor = System.Drawing.Color.FromArgb(newR, newG, newB);
                        bPic.SetPixel(i, j, newColor);
                        index++;
                    }
                    if (st)
                    {
                        break;
                    }
                }
                //pictureBox1.Image = bPic;


                //картинка со скрытым сообщением
```

```
                string sFilePic = directoryPath + "//bmpwithtext.bmp";
                FileStream wFile;
                try
                {
                    wFile = new FileStream(sFilePic, FileMode.Create); //открываем поток на запись
результатов
                }
                catch (IOException)
                {
                    MessageBox.Show("Ошибка открытия файла на запись", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    return;
                }

                bPic.Save(wFile, System.Drawing.Imaging.ImageFormat.Bmp);

                wFile.Close(); //закрываем поток   -   -------------------



                Aspose.Pdf.Document pdfDocument = new Aspose.Pdf.Document(pdf_filename);

                FileStream strim_pdf = new FileStream(sFilePic, FileMode.Open);
                try
                {
                    pdfDocument.Pages[1].Resources.Images.Replace(1, strim_pdf);//;
                    pdfDocument.Save(pdf_filename);
                    MessageBox.Show("Сообщение скрыто", "Информация");
                    //wFile = new FileStream(sFilePic, FileMode.Create); //открываем поток на запись
результатов
                }
                catch (IOException)
                {
                    MessageBox.Show("Ошибка открытия файла на запись", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                    return;
                }
                strim_pdf.Close();
                Pas1.Text = "";
                Pas2.Text = "";


            }

        }


        private void Decrypt_Click(object sender, EventArgs e)
        {
            string FilePic = directoryPath + "//bmpwithtext.bmp";
            FileStream rFile;
            try
            {
                rFile = new FileStream(FilePic, FileMode.Open); //открываем поток
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                return;
            }
            Bitmap bPic = new Bitmap(rFile);
            if (!isEncryption(bPic))
            {
                MessageBox.Show("В файле нет зашифрованной информации", "Информация",
MessageBoxButtons.OK);
                rFile.Close();
                return;
            }

            int countSymbol = ReadCountText(bPic); //считали количество зашифрованных символов
            byte[] message = new byte[countSymbol];
            int index = 0;
            bool st = false;
            for (int i = ENCRYP_TEXT_SIZE + 1; i < bPic.Width; i++)
            {
```

```
                for (int j = 0; j < bPic.Height; j++)
                {
                    System.Drawing.Color pixelColor = bPic.GetPixel(i, j);
                    if (index == message.Length)
                    {
                        st = true;
                        break;
                    }
                    BitArray colorArray = ByteToBit(pixelColor.R);
                    BitArray messageArray = ByteToBit(pixelColor.R); ;
                    messageArray[0] = colorArray[0];
                    messageArray[1] = colorArray[1];

                    colorArray = ByteToBit(pixelColor.G);
                    messageArray[2] = colorArray[0];
                    messageArray[3] = colorArray[1];
                    messageArray[4] = colorArray[2];

                    colorArray = ByteToBit(pixelColor.B);
                    messageArray[5] = colorArray[0];
                    messageArray[6] = colorArray[1];
                    messageArray[7] = colorArray[2];
                    message[index] = BitToByte(messageArray);
                    index++;
                }
                if (st)
                {
                    break;
                }
            }
            string strMessage = Encoding.GetEncoding(1251).GetString(message);



            if (!is_shifred) { // если файл зашифрован
                FileStream wFile; string sFileText = directoryPath + "//" +
"Извлеченная_информация.txt";
            try
            {
                wFile = new FileStream(sFileText, FileMode.Create); //открываем поток на запись
результатов
            }
            catch (IOException)
            {
                MessageBox.Show("Ошибка открытия файла на запись", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
                rFile.Close();
                return;
            }


            StreamWriter wText = new StreamWriter(wFile, Encoding.Default);
            //wText.Write(Decode_perestanovka(strMessage, key_str));
            wText.Write(strMessage);
            MessageBox.Show("Текст записан в файл Извлеченная_информация.txt", "Информация",
MessageBoxButtons.OK);
            wText.Close();
            wFile.Close(); //закрываем поток

            }


            else
            {
                {
                    if ((password == P3.Text) && (is_shifred))
                    {
                        FileStream wFile; string sFileText = directoryPath + "//" +
"Извлеченный_зашифрованный.txt";
                        try
                        {
                            wFile = new FileStream(sFileText, FileMode.Create); //открываем поток на
запись результатов
                        }
                        catch (IOException)
```

```
                        {
                            MessageBox.Show("Ошибка открытия файла на запись", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                            rFile.Close();
                            return;
                        }


                        StreamWriter wText = new StreamWriter(wFile, Encoding.Default);
                        //wText.Write(Decode_perestanovka(strMessage, key_str));
                        wText.Write(strMessage);
                        MessageBox.Show("Текст записан в файл Извлеченная_информация.txt", "Информация",
MessageBoxButtons.OK);
                        wText.Close();
                        wFile.Close(); //закрываем поток




                        StreamReader f3 = new StreamReader((directoryPath + "//" +
"Извлеченный_зашифрованный.txt"));
                        FileStream file3 = new FileStream(directoryPath + "//" +
"Извлеченная_информация.txt", FileMode.Create);
                        StreamWriter fnew3 = new StreamWriter(file3);
                        while (!f3.EndOfStream)
                        {


                            string encrypt_pass = f3.ReadLine();
                            string key_sort = string.Join("", password);//key;
                            string decode = Decode_perestanovka(encrypt_pass, key_str); // string
decoded_pass = Decode( encrypt_pass , key_sort); //подставляем сортированный ключ то есть логин
                            fnew3.WriteLine(decode);

                        }
                        f3.Close();
                        fnew3.Close();
                    }
                    else { MessageBox.Show("Неправильный пароль", "Ошибка"); }
                }



                rFile.Close();
            }

        }

        private void checkBox1_CheckedChanged(object sender, EventArgs e)
        {
            is_shifred = !is_shifred;
        }
    }




}
```