САНКТ-­ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра КБ

КУРСОВОЙ ПРОЕКТ

по дисциплине «Программирование»

Тема: Стеганография в изображении

Выполнил:   Нерсисян Артур

Группа:  6361

Факультет:  КТИ

Проверил:  Пелевин М.С.

Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: «  » декабря 2016 г.

Санкт-­Петербург

2016

Содержимое

История--------------------------------------------------------------------------------------3

Компьютерная стеганография-------------------------------------------------------3

Цифровая стеганография-------------------------------------------------------------4

Алгоритмы и методы--------------------------------------------------------------------4

Метод LSB----------------------------------------------------------------------------------5

Атаки на стегосистемы-----------------------------------------------------------------6

Структура BMP файла------------------------------------------------------------------7

Практическая реализация------------------------------------------------------------10

Шифрование/ расшифровка---------------------------------------------------------11

Работа с файлами----------------------------------------------------------------------11

Проблемы----------------------------------------------------------------------------------11

Работа с программой------------------------------------------------------------------12

Заключение-------------------------------------------------------------------------------13

Список используемой литературы-------------------------------------------------13

История

**Стеганография** (от греч. στεγανός — скрытый + γράφω — пишу; буквально «**тайнопись**») — способ передачи или хранения информации с учётом **сохранения в тайне самого факта такой передачи** (хранения). Этот термин ввел в 1499 году Иоганн Тритемий в своем трактате «**Стеганография**» (*Steganographia*), зашифрованном под магическую книгу.

В отличие от [**криптографии**](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B8%D0%BF%D1%82%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F), которая скрывает содержимое тайного сообщения, **стеганография скрывает сам факт его существования**. Как правило, сообщение будет выглядеть как что-либо иное, например, как изображение, статья, список покупок, письмо или судоку. Стеганографию обычно используют совместно с методами криптографии, таким образом, дополняя её.

**Стеганографическая система** (стегосистема) — объединение методов и средств используемых для создания скрытого канала для передачи информации. Стеганография в наше время часто применяется, как правило, **для встраивания цифровых водяных знаков**, являющееся **основой для систем защиты авторских прав и DRM** (Digital rights management) систем.

Существует несколько классификаций стеганографии

* **Классическая стеганография**
* **Компьютерная стеганография**
* **Цифровая стеганография**

Компьютерная стеганография

**Компьютерная стеганография** — направление классической стеганографии, основанное на особенностях компьютерной платформы. Примеры — стеганографическая файловая система StegFS для Linux, скрытие данных в неиспользуемых областях форматов файлов, подмена символов в названиях файлов, [текстовая стеганография](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D0%B0%D1%8F_%D1%81%D1%82%D0%B5%D0%B3%D0%B0%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F) и т. д.

Приведём некоторые примеры:

Использование зарезервированных полей компьютерных [форматов файлов](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BC%D0%B0%D1%82_%D1%84%D0%B0%D0%B9%D0%BB%D0%B0) — суть метода состоит в том, что часть [поля расширений](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D1%81%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%B8%D0%B5_%D1%84%D0%B0%D0%B9%D0%BB%D0%B0), не заполненная информацией о расширении, по умолчанию заполняется нулями. Соответственно мы можем использовать эту «нулевую» часть для записи своих данных. Недостатком этого метода является низкая степень скрытности и малый объём передаваемой информации.

Метод скрытия информации в неиспользуемых местах [гибких дисков](https://ru.wikipedia.org/wiki/%D0%93%D0%B8%D0%B1%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA) — при использовании этого метода информация записывается в неиспользуемые части диска, к примеру, на нулевую дорожку. Недостатки: маленькая производительность, передача небольших по объёму сообщений.

Метод использования особых свойств полей форматов, которые не отображаются на экране — этот метод основан на специальных «невидимых» полях для получения сносок, указателей. К примеру, написание чёрным шрифтом на чёрном фоне. Недостатки: маленькая производительность, небольшой объём передаваемой информации.

Использование особенностей файловых систем — при хранении на жёстком диске файл всегда (не считая некоторых ФС, например, ReiserFS) занимает целое число кластеров (минимальных адресуемых объёмов информации). К примеру, в ранее широко используемой файловой системе FAT32 (использовалась в Windows98/Me/2000) стандартный размер кластера — 4 КБ. Соответственно для хранения 1 КБ информации на диске выделяется 4 КБ памяти, из которых 1 КБ нужен для хранения сохраняемого файла, а остальные 3 ни на что не используются — соответственно их можно использовать для хранения информации. Недостаток данного метода: лёгкость обнаружения.

Цифровая стеганография

**Цифровая стеганография** — направление классической стеганографии, основанное на сокрытии или внедрении дополнительной информации в цифровые объекты, вызывая при этом некоторые искажения этих объектов. Но, как правило, данные объекты являются мультимедиа-объектами (изображения, видео, аудио, текстуры 3D-объектов) и внесение искажений, которые находятся ниже порога чувствительности среднестатистического человека, не приводит к заметным изменениям этих объектов. Кроме того, в оцифрованных объектах, изначально имеющих аналоговую природу, всегда присутствует шум квантования; далее, при воспроизведении этих объектов появляется дополнительный аналоговый шум и нелинейные искажения аппаратуры, все это способствует большей незаметности сокрытой информации.

Алгоритмы и методы

Все алгоритмы встраивания скрытой информации можно разделить на несколько подгрупп:

* Работающие с самим цифровым сигналом. Например, метод LSB.
* «Впаивание» скрытой информации. В данном случае происходит наложение скрываемого изображения (звука, иногда текста) поверх оригинала. Часто используется для встраивания цифровых водяных знаков (ЦВЗ).
* Использование особенностей форматов файлов. Сюда можно отнести запись информации в метаданные или в различные другие не используемые зарезервированные поля файла.

Методы:

* Метод LSB
* Метод изменения величин дискретного косинусного преобразования (ДКП)
* Эхо-методы
* Фазовое кодирование
* [Метод расширенного спектра](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%B3%D0%B0%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F#.D0.9C.D0.B5.D1.82.D0.BE.D0.B4_.D1.80.D0.B0.D1.81.D1.88.D0.B8.D1.80.D0.B5.D0.BD.D0.BD.D0.BE.D0.B3.D0.BE_.D1.81.D0.BF.D0.B5.D0.BA.D1.82.D1.80.D0.B0)
* Широкополосные методы
* Статические методы
* Методы искажения
* Структурный метод

В данном проекте мы будем рассматривать только **Метод LSB**

Метод LSB

**LSB** (Least Significant Bit, наименьший значащий бит (НЗБ)) — суть этого метода заключается в замене последних значащих битов в контейнере (изображения, аудио или видеозаписи) на биты скрываемого сообщения. Разница между пустым и заполненным контейнерами должна быть не ощутима для органов восприятия человека.

Суть метода заключается в следующем: Допустим, имеется 8-битное изображение в градациях серого. 00h (00000000b) обозначает чёрный цвет,

FFh (11111111b) — белый. Всего имеется 256 градаций ({\displaystyle 2^{8}}28). Также предположим, что сообщение состоит из 1 байта — например, 01101011b. При использовании 2 младших бит в описаниях пикселей, нам потребуется 4 пикселя. Допустим, они чёрного цвета. Тогда пиксели, содержащие скрытое сообщение, будут выглядеть следующим образом: 00000001 00000010 00000010 00000011. Тогда цвет пикселей изменится: первого — на 1/255, второго и третьего — на 2/255 и четвёртого — на 3/255. Такие градации, мало того, что незаметны для человека, могут вообще не отобразиться при использовании низкокачественных устройств вывода.

Методы LSB являются неустойчивыми ко всем видам атак и могут быть использованы только при отсутствии шума в канале передачи данных.

Обнаружение LSB-кодированного стего осуществляется по аномальным характеристикам распределения значений диапазона младших битов отсчётов цифрового сигнала.

Все методы LSB являются, как правило, аддитивными (A17, L18D).

Другие методы скрытия информации в графических файлах ориентированы на форматы файлов с потерей, к примеру, JPEG. В отличие от LSB они более устойчивы к геометрическим преобразованиям. Это получается за счёт варьирования в широком диапазоне качества изображения, что приводит к невозможности определения источника изображения.

В данном проекте будем использовать метод, похожий методу LSB.

Будем поступать следующим образом:

В один пиксель изображения будем шифровать один символ текста методом 2-3-3, то есть:

1 пиксель изображения в формате BMP-24 занимает 3 байта, а 1 символ текста 8 бит (1 байт), так как пиксель изображения состоит из трех каналов RGB (Red Green Blue) на каждый канал цвета соответственно нужно кодировать треть от байта текста таким образом:

Пиксель до кодировки символ текста Пиксель после кодировки

R: 111100**10** R: 111100**10**   
G: 11000**101 🡪 10111001 🡪** G: 11000**111 🡪**   
B: 11001**010** B: 11001**001**

Атаки на стегосистемы

Под атакой на стегосистему понимается попытка обнаружить, извлечь, изменить скрытое стеганографическое сообщение. Такие атаки называются стегоанализом по аналогии с криптоанализом для криптографии. Способность стеганографической системы противостоять атакам называется стеганографической стойкостью. Наиболее простая атака — субъективная. Внимательно рассматривается изображение, прослушивается звукозапись в попытках найти признаки существования в нём скрытого сообщения. Такая атака имеет успех лишь для совсем незащищенных стегосистем. Обычно это первый этап при вскрытии стегосистемы. Выделяются следующие типы атак.

* Атака по известному заполненному контейнеру;
* Атака по известному встроенному сообщению;
* Атака на основе выбранного скрытого сообщения;
* Адаптивная атака на основе выбранного скрытого сообщения;
* Атака на основе выбранного заполненного контейнера;
* Атака на основе известного пустого контейнера;
* Атака на основе выбранного пустого контейнера;
* Атака по известной математической модели контейнера.

Структура BMP файла

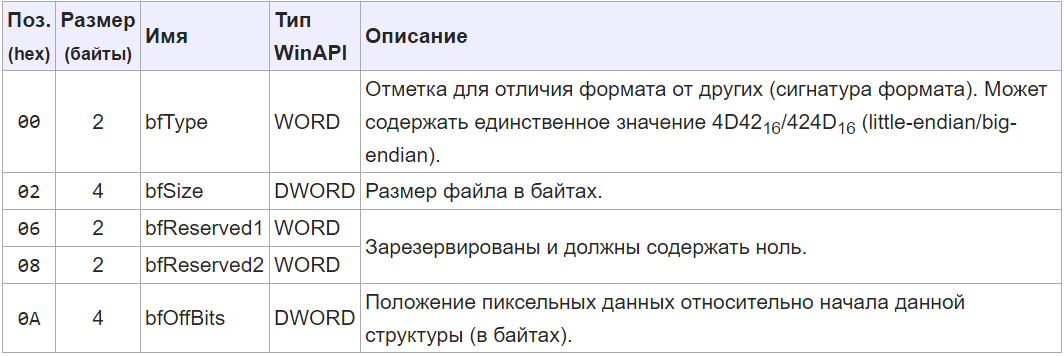
**BMP** (от англ. *Bitmap Picture*) — формат хранения растровых изображений, разработанный компанией Microsoft. Файлы формата BMP могут иметь расширения .bmp, .dib и .rle.

Данные в формате BMP состоят из трёх основных блоков различного размера:

1. Заголовок из структуры BITMAPFILEHEADER и блока BITMAPINFO. Последний содержит:
   * 1. Информационные поля.
     2. Битовые маски для извлечения значений цветовых каналов (опциональные).
     3. Таблица цветов (опциональная).
2. Цветовой профиль (опциональный).
3. Пиксельные данные.

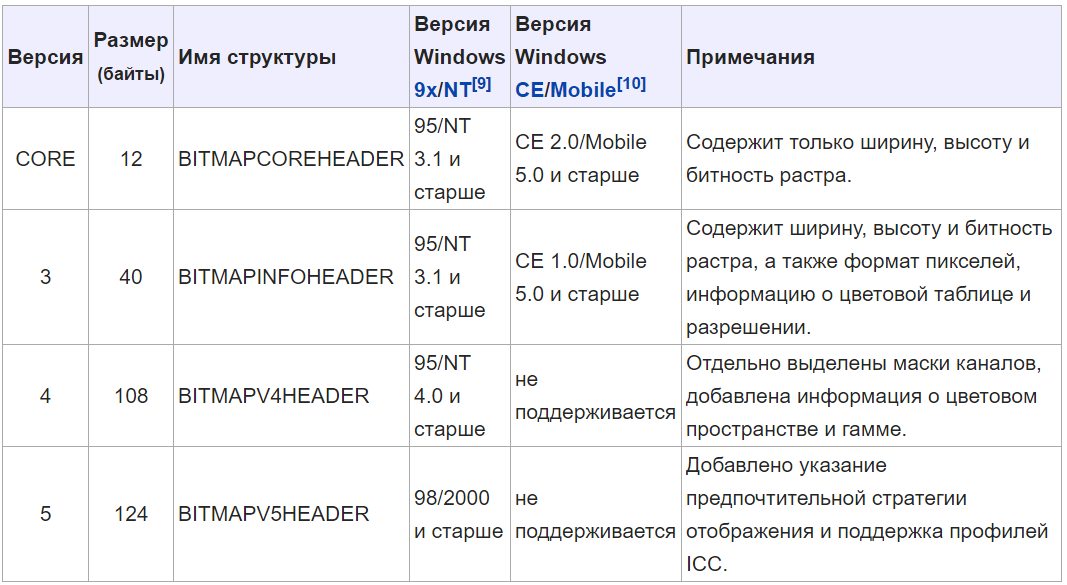
BITMAPFILEHEADER

BITMAPFILEHEADER — 14-байтная структура, которая располагается в самом начале файла. Обратите внимание на то, что с самого начала структуры сбивается выравнивание ячеек. Если для вас оно важно, то в оперативной памяти данный заголовок располагайте по чётным адресам, которые не кратны четырём (тогда 32-битные ячейки попадут на выравненные позиции).

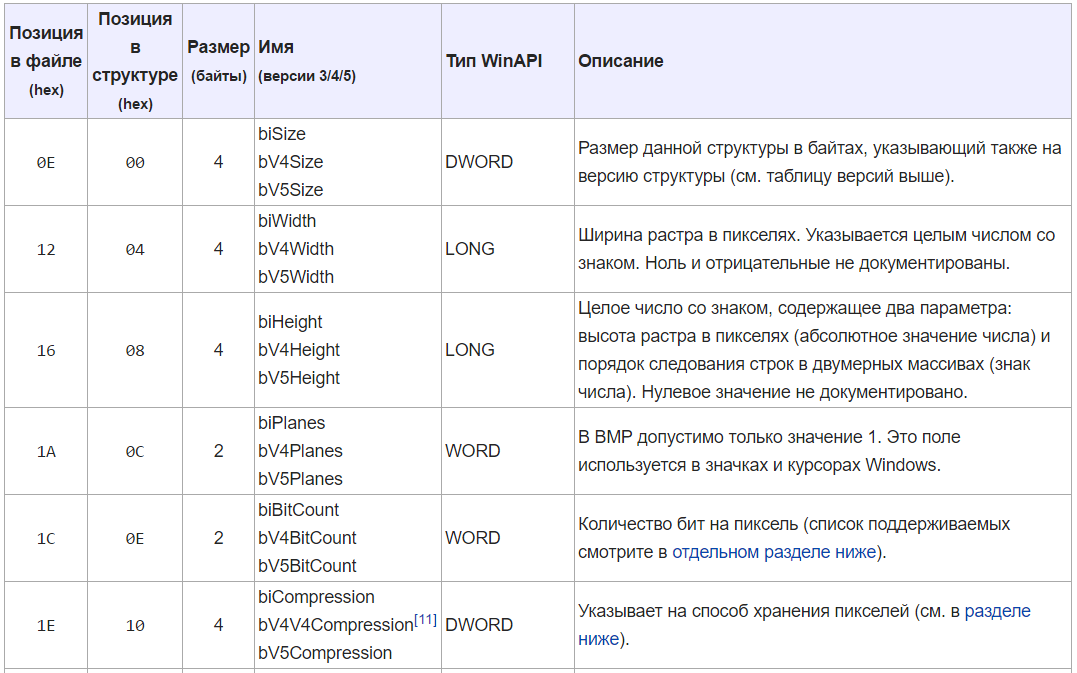


### BITMAPINFO

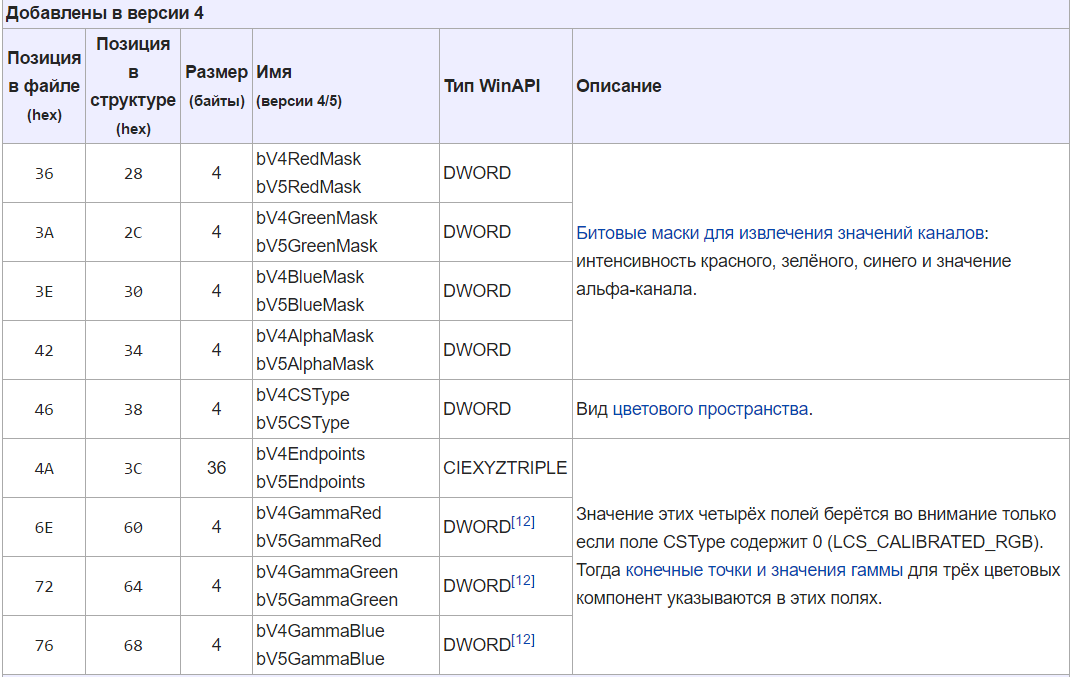
BITMAPINFO в файле идёт сразу за BITMAPFILEHEADER. Адрес этого блока в памяти напрямую также передаётся некоторым функциям WinAPI (например, SetDIBitsToDevice или CreateDIBitmap). Кроме этого, этот же блок используется в форматах значков и курсоров Windows, но в данной статье этот момент не рассматривается (см. отдельные описания этих форматов). Данная структура является основной и описательной в формате BMP и поэтому когда просто упомянуто имя поля, то речь идёт о поле в данной структуре.



#### 32-битные информационные поля (версии 3, 4 и 5)







Практическая реализация

Для сохранения параметров, заголовка файла, разрешении изображении и остальной информации об изображении в программе используется следующая структура:

unsigned char b1, b2;

unsigned long bfSize;

unsigned short bfReserved1;

unsigned short bfReserved2;

unsigned long bfOffBits;

unsigned int biSize;

int biWidth;

int biHeight;

unsigned short biPlanes;

unsigned short biBitCount;

unsigned int biCompression;

unsigned int biSizeImage;

int biXPelsPerMeter;

int biYPelsPerMeter;

unsigned int biClrUsed;

unsigned int biClrImportant;

Поля выравниваются в памяти по границе кратной своему же размеру (1-байтовые поля не выравниваются, 2-байтовые — выравниваются на чётные позиции, 4-байтовые — на позиции кратные четырём и т.д. В большинстве случаев выравнивание размера структуры в памяти составляет 4 байта.

Если в случае обычной структуры все поля внутри неё выравниваются на выравнивание базового типа поля, то при наличии ***#pragma pack*** это выравнивание ограничивается. В случае единицы все поля структуры будут идти впритык друг к другу без выравниваний, что позволит избежать ошибок в работе программы.

Шифрование/ расшифровка

Для большей надежности, в программе также используется шифр Цезаря. В случаи атаки кроме бессмысленного ряда букв и символов злоумышленник ничего не получит \*(надеюсь).

Само шифрование проходит следующим образом:

* Обнуляем последние биты каналов

currentLine[j].RGB = currentLine[j].RGB >> а;

currentLine[j].RGB = currentLine[j].RGB << а;

где а 2-3-3 соответсвенно.

* Добавляем биты текста по схеме 2-3-3

Расшифровка аналогичным методом – взимается последние биты по схеме 2-3-3 соединяем и получив символ записываем в буфер.

Работа с файлами

Для считывания и записи файлов используется класс ***fstream,*** чтобы оно работало подключаем библиотеку fstream *(#include <fstream>).* Также, чтобы было бы возможно работать с самими байтами файла, открываем файл в бинарном режиме.

fstream bmpfile;

bmpfile.open(bmpfilepath.c\_str(), ios::binary | ios::in);

Проблемы

В основном проблемы были

* Синтаксические ошибки
* Некорректная работа компилятора
* Компилятор не распознал некоторые функции
* Неполноценное знание языка
* Работа с { int argc char \*argv[] }
* Чрезмерное возрастание размера изображения (весь текст забивался в один пиксель, из-за чего битность картинки возросла, в следствии, общий размер тоже)

Работа с программой

Для работы программы нужно ввести:

$ program [textfile] <Option> [bmpfile] <key>

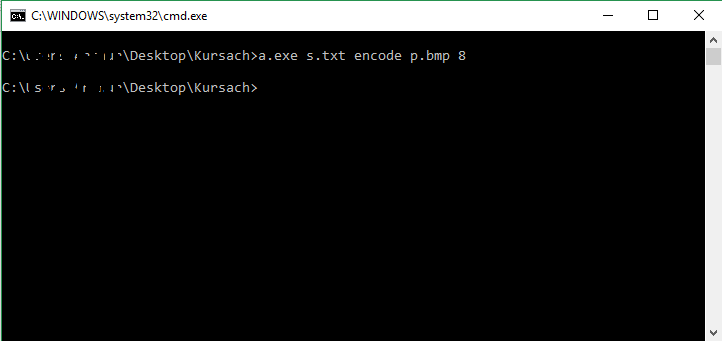
Option: type может быть “encode” или “decode”

Key: ключ для шифрования/дешифрования

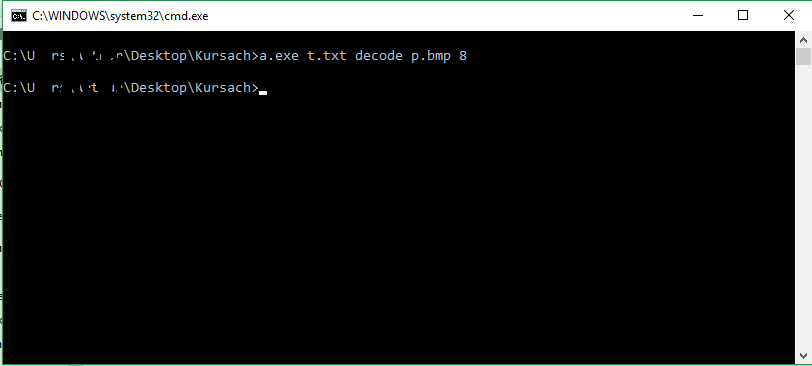
textfile: путь к txt файлу

bmpfile: путь к bmp файлу

*encoding*

****

*decoding*

****

Заключение

Программа работает, но все же не устойчива к атаке. При желании можно взломать, но преимущества этого метода в том, что никто даже не будет знать про существования скрытого сообщения. Способ идеален, но до того, как будет обнаружен. Во избежание подозрений о наличии скрытого сообщения, можно использовать криптографию, а потом стеганографию. Таким образом, если даже кто-то заподозрит, после нескольких попыток расшифровывать получит ряд бессмысленных символов и все. Для меньшего влияния на цвета в картинке можно использовать только последний бит из каждого канала цвета.

Почему BMP?

BMP – несжатый формат изображения и при 24-бит на пиксель (8-бит на канал) получаем больше место для текста. Формат jpeg (jpe, jpg, jpj, и т.д.) сжатый формат (8 бит на пиксель) и использовать для стеганографии не удобно (можно, но не удобно).

Список используемой литературы

<https://habrahabr.ru/post/140373/>

<http://www.cyberforum.ru/cpp-builder/thread101901.html>

<http://www.cyberforum.ru/cpp-builder/thread103448.html>

<https://ru.wikipedia.org/wiki/BMP>

[https://ru.wikipedia.org/wiki/**Стеганография**](https://ru.wikipedia.org/wiki/Стеганография)

книга C++ за 21 дней ( <http://cppstudio.com/post/3200/> )