

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**, 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 sudoku. 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 и т. д.

Приведём некоторые примеры:

Использование зарезервированных полей компьютерных форматов файлов — суть метода состоит в том, что часть поля\_расширений, 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 — 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 сносков, 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 Windows 98/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
- 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 ( $2^8$ ). 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: 11110010		R: 11110010
G: 11000101 →	10111001 →	G: 11000111 →
B: 11001010		B: 11001001

## 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 это важно, то в оперативной памяти данный заголовок располагайте по чётным адресам, которые не кратны четырём (тогда 32-битные ячейки попадут на выровненные позиции).

Поз. (hex)	Размер (байты)	Имя	Тип WinAPI	Описание
00	2	bfType	WORD	Отметка для отличия формата от других (сигнатура формата). Может содержать единственное значение <code>4D42<sub>16</sub>/424D<sub>16</sub></code> (little-endian/big-endian).
02	4	bfSize	DWORD	Размер файла в байтах.
06	2	bfReserved1	WORD	Зарезервированы и должны содержать ноль.
08	2	bfReserved2	WORD	
0A	4	bfOffBits	DWORD	Положение пиксельных данных относительно начала данной структуры (в байтах).

## 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.

Версия	Размер (байты)	Имя структуры	Версия Windows 9x/NT <sup>[9]</sup>	Версия Windows CE/Mobile <sup>[10]</sup>	Примечания
CORE	12	BITMAPCOREHEADER	95/NT 3.1 и старше	CE 2.0/Mobile 5.0 и старше	Содержит только ширину, высоту и битность раstra.
3	40	BITMAPINFOHEADER	95/NT 3.1 и старше	CE 1.0/Mobile 5.0 и старше	Содержит ширину, высоту и битность раstra, а также формат пикселей, информацию о цветовой таблице и разрешении.
4	108	BITMAPV4HEADER	95/NT 4.0 и старше	не поддерживается	Отдельно выделены маски каналов, добавлена информация о цветовом пространстве и гамме.
5	124	BITMAPV5HEADER	98/2000 и старше	не поддерживается	Добавлено указание предпочтительной стратегии отображения и поддержка профилей ICC.

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

Позиция в файле (hex)	Позиция в структуре (hex)	Размер (байты)	Имя (версии 3/4/5)	Тип WinAPI	Описание
0E	00	4	biSize bV4Size bV5Size	DWORD	Размер данной структуры в байтах, указывающий также на версию структуры (см. таблицу версий выше).
12	04	4	biWidth bV4Width bV5Width	LONG	Ширина раstra в пикселях. Указывается целым числом со знаком. Ноль и отрицательные не документированы.
16	08	4	biHeight bV4Height bV5Height	LONG	Целое число со знаком, содержащее два параметра: высота раstra в пикселях (абсолютное значение числа) и порядок следования строк в двумерных массивах (знак числа). Нулевое значение не документировано.
1A	0C	2	biPlanes bV4Planes bV5Planes	WORD	В BMP допустимо только значение 1. Это поле используется в значках и курсорах Windows.
1C	0E	2	biBitCount bV4BitCount bV5BitCount	WORD	Количество бит на пиксель (список поддерживаемых смотрите в <a href="#">отдельном разделе ниже</a> ).
1E	10	4	biCompression bV4V4Compression <sup>[11]</sup> bV5Compression	DWORD	Указывает на способ хранения пикселей (см. в <a href="#">разделе ниже</a> ).



22	14	4	biSizeImage bV4SizeImage bV5SizeImage	DWORD	Размер пиксельных данных в байтах. Может быть обнулено если хранение осуществляется двумерным массивом.
26	18	4	biXPelsPerMeter bV4XPelsPerMeter bV5XPelsPerMeter	LONG	Количество пикселей на метр по горизонтали и вертикали (см. раздел «Разрешение изображения» данной статьи).
2A	1C	4	biYPelsPerMeter bV4YPelsPerMeter bV5YPelsPerMeter	LONG	
2E	20	4	biClrUsed bV4ClrUsed bV5ClrUsed	DWORD	Размер <a href="#">таблицы цветов</a> в ячейках.
32	24	4	biClrImportant bV4ClrImportant bV5ClrImportant	DWORD	Количество ячеек от начала таблицы цветов до последней используемой (включая её саму).

#### Добавлены в версии 4

Позиция в файле (hex)	Позиция в структуре (hex)	Размер (байты)	Имя (версии 4/5)	Тип WinAPI	Описание
36	28	4	bV4RedMask bV5RedMask	DWORD	<b>Битовые маски для извлечения значений каналов:</b> интенсивность красного, зелёного, синего и значение альфа-канала.
3A	2C	4	bV4GreenMask bV5GreenMask	DWORD	
3E	30	4	bV4BlueMask bV5BlueMask	DWORD	
42	34	4	bV4AlphaMask bV5AlphaMask	DWORD	
46	38	4	bV4CSType bV5CSType	DWORD	Вид <a href="#">цветового пространства</a> .
4A	3C	36	bV4Endpoints bV5Endpoints	CIXYZTRIPLE	Значение этих четырёх полей берётся во внимание только если поле CSType содержит 0 (LCS_CALIBRATED_RGB). Тогда <b>конечные точки и значения гаммы</b> для трёх цветовых компонент указываются в этих полях.
6E	60	4	bV4GammaRed bV5GammaRed	DWORD <sup>[12]</sup>	
72	64	4	bV4GammaGreen bV5GammaGreen	DWORD <sup>[12]</sup>	
76	68	4	bV4GammaBlue bV5GammaBlue	DWORD <sup>[12]</sup>	

#### Добавлены в версии 5

Позиция в файле (hex)	Позиция в структуре (hex)	Размер (байты)	Имя	Тип WinAPI	Описание
7A	6C	4	bV5Intent	DWORD	<a href="#">Предпочтения при рендеринге</a> раstra.
7E	70	4	bV5ProfileData	DWORD	Смещение в байтах цветового профиля от начала BITMAPINFO (см. также раздел « <a href="#">Цветовой профиль</a> » ниже в этой статье).
82	74	4	bV5ProfileSize	DWORD	Если в BMP непосредственно включается цветовой профиль, то здесь указывается его размер в байтах.
86	78	4	bV5Reserved	DWORD	Зарезервировано и должно быть обнулено.

## Практическая реализация

Для сохранения параметров, заголовка файла, разрешения изображения 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 случае 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 >> a;`  
`currentLine[j].RGB = currentLine[j].RGB << a;`  
где a 2-3-3 соответственно.
- 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>
```

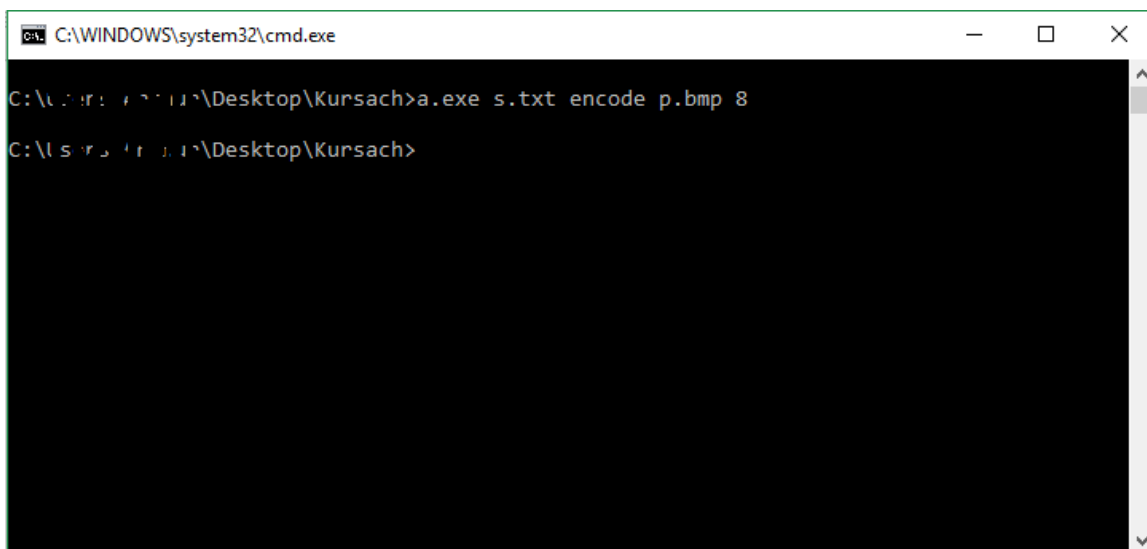
Option: type может быть “encode” или “decode”

Key: ключ для шифрования/дешифрования

textfile: путь к txt файлу

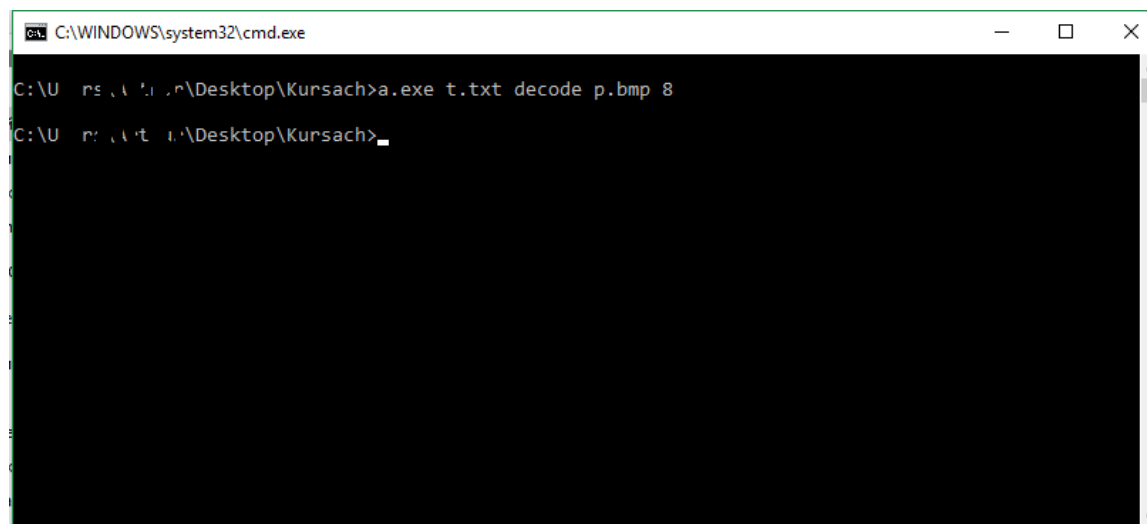
bmpfile: путь к bmp файлу

*encoding*



```
C:\WINDOWS\system32\cmd.exe
C:\Users\user\Desktop\Kursach>a.exe s.txt encode p.bmp 8
C:\Users\user\Desktop\Kursach>
```

*decoding*



```
C:\WINDOWS\system32\cmd.exe
C:\Users\user\Desktop\Kursach>a.exe t.txt decode p.bmp 8
C:\Users\user\Desktop\Kursach>
```

## 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 криптография, а потом стеганографию. Таким образом, если даже кто-то заподозрит, после нескольких попыток расшифровывать получит ряд 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, jpeg, 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>

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