1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский политехнический университет Петра Великого
3. —
4. Институт компьютерных наук и технологий
5. **Кафедра «Информационная безопасность компьютерных систем»**

**ОТЧЕТ**

**по лабораторной работе № 6**

По дисциплине «Основы информационной безопасности»

1. Выполнил
2. Студент гр. 13508/13 А.Э.Палёный

\_\_\_\_\_\_\_\_\_\_

1. Проверил
2. Преподаватель

\_\_\_\_\_\_\_\_\_\_ М.О.Калинин

1. Санкт-Петербург
2. 2016

Цель

Приобретение навыков исследования свойств стегоконтейнеров, разработки стегосистем и их применения для сокрытия данных при передаче с помощью графических изображений.

Решаемые задачи

1. Проверить наличие палитры в BMP изображении
2. Создать программу, которая будет скрывать текст в изображении
3. Создать программу, которая будет расшифровывать закодированный текст в изображении

Содержание:

Ход работы (4-6)

Листинг кодера (6-14)

Листинг декодера (14-21)

Результаты влияния степеней упаковки на полученное изображение (21-22)

Ответы на контрольные вопросы (22-24)

Вывод (24)

Ход работы

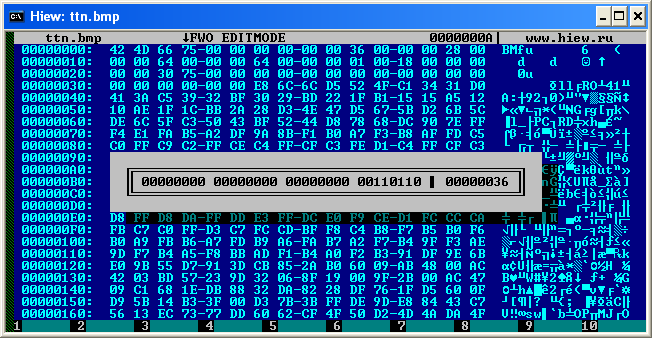
BMP (от англ. Bitmap Picture) — формат хранения растровых изображений, разработанный компанией Microsoft.

BMP представляет из себя несжатое (в основном) изображение, которое довольно легко читается и выводится в ОС Windows, в которой есть специальные функции API, которые в этом помогают.

Рассмотрим его структуру:

|  |  |
| --- | --- |
| **Структура файла BMP** | |
| **Заголовок файла растровой графики (14 байт)** | |
| Сигнатура файла BMP (2 байт) | Обычно код 4D42h - Буквы 'BM' |
| Размер файла (4 байт) | В unsigned int |
| Не используется (2 байт) | Резервные поля. Обычно не используются |
| Не используется (2 байт) |
| Смещение, с которого начинается само изображение (4 байт) | В unsigned int |
| **Информационный заголовок растрового массива (40 байт)** | |
| Длина этого заголовка (4 байт) | В unsigned int |
| Ширина изображения (4 байт) | В unsigned int |
| Высота изображения (4 байт) | В unsigned int |
| Число цветовых плоскостей (2 байт) | В unsigned short. Число плоскостей, должно быть 1 |
| Бит/пиксель (2 байт) | Бит/пиксел: 1, 4, 8 или 24 |
| Метод сжатия (4 байт) | Тип сжатия(0 по умолчанию) |
| Длина растрового массива (4 байт) | 0 или размер изображения в байтах. |
| Горизонтальное разрешение (4 байт) | В unsigned int. Пиксел/м |
| Вертикальное разрешение (4 байт) | В unsigned int. Пиксел/м |
| Число цветов изображения (4 байт) | Количество используемых цветов (0 по умолчанию) |
| Число основных цветов (4 байт) | Количество "важных" цветов. |
| **Таблица цветов (длина изменяется от 8 до 1024 байт)(Обычно не используется)** | |
| **Собственно данные растрового массива (длина переменная)** | |

2) Исходный размер файла BMP равен - 315 KB (323 526 bytes).  
Палитры нету, т.к



В Hex редакторе показано, что смещение равно 0x00000036. Переведём это в десятичную систему исчисления. Результат равен 54. Следовательно, на палитру место не остаётся, следовательно её нету.

3) Размер содержательной части равен ширина\*высота\*3. Т.е 30000 байт в данном случае.

1. Расчёт максимальной длины.  
   Макс. Длина = (Содержат. Часть – 9\*8/(количество изменённых бит на байт))8/ количество изменённых бит на байт

**Листинг Кодера**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <malloc.h>

typedef struct ConBMP

{

//File Info

unsigned short bType;

unsigned int bSize;

unsigned int bReserved;

unsigned int bOffset;

//Image Info

unsigned int iSize;

unsigned int bWidth;

unsigned int bHeight;

unsigned short bPlans;

unsigned short BitPixel;

unsigned int bCompress;

unsigned int SizeCoImage;

unsigned int bXPixel;

unsigned int bYPixel;

unsigned int bClrUsed;

unsigned int bClrImportant;

unsigned int bPalitra;

} BMPheader;

typedef unsigned char byte;//Change the "Unsigned Char" to "byte"

byte \*LongToBit(unsigned long long data)

{

int i;

int Tsize=sizeof(unsigned long long)\*8;

byte \*res=(byte\*)malloc(Tsize+1);

for (i=0; i < Tsize; i++)

{

if ((data >> i & 1) == 1) res[i]=1;

else res[i]=0;

}

return res;

}

byte \*ByteToBit(byte data)

{

int i;

byte \*res=(byte\*)malloc(8+1);

for (i=0; i < 8; i++)

{

if ((data >> i & 1) == 1) res[i]=1;

else res[i]=0;

}

return res;

}

byte BitToByte(byte \*data)

{

int i;

byte res=0;

for (i=0; i < 8; i++)

if(data[i] == 1)

res+=pow(2.0, i);

return res;

}

BMPheader ReadHeader(FILE \*Input)

{

BMPheader Img;

fread( &Img.bType, 1, sizeof(short), Input);

fread( &Img.bSize, 1, sizeof(int), Input);

fread( &Img.bReserved, 1, sizeof(int), Input);

fread( &Img.bOffset, 1, sizeof(int), Input);

fread( &Img.iSize, 1, sizeof(int), Input);

fread( &Img.bWidth, 1, sizeof(int), Input);

fread( &Img.bHeight, 1, sizeof(int), Input);

fread( &Img.bPlans, 1, sizeof(short), Input);

fread( &Img.BitPixel, 1, sizeof(short), Input);

fread( &Img.bCompress, 1, sizeof(int), Input);

fread( &Img.SizeCoImage, 1, sizeof(int), Input);

fread( &Img.bXPixel, 1, sizeof(int), Input);

fread( &Img.bYPixel, 1, sizeof(int), Input);

fread( &Img.bClrUsed, 1, sizeof(int), Input);

fread( &Img.bClrImportant, 1, sizeof(int), Input);

if(Img.bOffset == 54) Img.bPalitra=0;

else Img.bPalitra=1;

return Img;

}

byte \*PutPixels(FILE \*Input, int \*count, BMPheader \*Img)

{

int i;

byte \*GetPixel;

\*Img=ReadHeader(Input);

i=Img->bHeight \* Img->bWidth\*3;

GetPixel=(byte\*)malloc(i+1);

\*count=i;

for(i; i!=0; i--)

fread(&GetPixel[i-1], 1, sizeof(byte), Input);

return GetPixel;

}

byte \*ReadFile(unsigned long long \*SCount)

{

char c;

unsigned long long FSize;

byte \*TextFile;

FILE \*Output=fopen("Output.txt", "rb+");

fseek(Output,0,SEEK\_END);

FSize = ftell(Output);

TextFile=(byte\*)malloc(FSize+1);

fseek(Output,0,SEEK\_SET);

for(FSize=0; fscanf(Output,"%c",&c) != EOF; FSize++)

TextFile[FSize]=c;

\*SCount=FSize;

fclose(Output);

return TextFile;

}

void Write(byte \*GetPixel, byte \*TextFile, unsigned long long FSize, int p)

{

int f;

int popravka=0;

int i=0;

int k=0;

int TPos=0;

unsigned char \*ImgDump, \*TxtDump;

TPos=sizeof(unsigned long long)\*8;

TxtDump=ByteToBit('T');

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

{

ImgDump=ByteToBit(GetPixel[i]);

if(k+p>7) popravka=k+p-8;

for(f=p-popravka; f>0; f--)

{

ImgDump[f]=TxtDump[k];

k++;

}

k--;

popravka=0;

GetPixel[i]=BitToByte(ImgDump);

free(ImgDump);

i++;

}

free(TxtDump);

TxtDump=LongToBit(FSize);

for(k=0;k<TPos;k++)

{

ImgDump=ByteToBit(GetPixel[i]);

if(k+p>TPos-1)

popravka=k+p-TPos;

for(f=p-popravka; f>0; f--)

{

ImgDump[f]=TxtDump[k];

k++;

}

k--;

popravka=0;

GetPixel[i]=BitToByte(ImgDump);

free(ImgDump);

i++;

}

free(TxtDump);

for(TPos=0; TPos!=FSize; TPos++)

{

TxtDump=ByteToBit(TextFile[TPos]);

for(k=0; k!=8; k++)

{

ImgDump=ByteToBit(GetPixel[i]);

if(k+p>7) popravka=k+p-8;

for(f=p-popravka; f>0; f--)

{

ImgDump[f]=TxtDump[k];

k++;

}

k--;

popravka=0;

GetPixel[i]=BitToByte(ImgDump);

free(ImgDump);

i++;

}

free(TxtDump);

}

}

int main()

{

int count,i,n;

float dump;

unsigned long long FSize;

FILE \*Input=fopen("ttn.bmp", "rb+");

byte \*TextFile;

byte \*GetPixel;

BMPheader Img;

n=0;

printf("Enter the stepen' sjatiya ot 1 do 8\n");

scanf("%d",&n);

if( !Input ) return NULL;

GetPixel=PutPixels(Input, &count, &Img);

TextFile=ReadFile(&FSize);

dump=(float) (9+FSize)\*8/n;

if(Img.bHeight\*Img.bWidth\*3 >= dump)

{

Write(GetPixel, TextFile,FSize, n);

fseek(Input, 54, SEEK\_SET);

for(i=count-1; i>=0; i--)

fwrite(&GetPixel[i],1,1,Input);

printf("\nDone");

}else printf("The file is too big");

fclose(Input);

free(GetPixel);

free(TextFile);

getchar();

getchar();

}

**Листинг декодера**

#include <stdio.h>

#include <math.h>

#include <stdlib.h>

typedef struct ConBMP

{

//File Info

unsigned short bType;

unsigned int bSize;

unsigned int bReserved;

unsigned int bOffset;

//Image Info

unsigned int iSize;

unsigned int bWidth;

unsigned int bHeight;

unsigned short bPlans;

unsigned short BitPixel;

unsigned int bCompress;

unsigned int SizeCoImage;

unsigned int bXPixel;

unsigned int bYPixel;

unsigned int bClrUsed;

unsigned int bClrImportant;

unsigned int bPalitra;

} BMPheader;

typedef unsigned char byte;//Change the "Unsigned Char" to "byte"

unsigned long long BitToLong(byte \*data)

{

int i;

//int TSize=sizeof(unsigned long long)\*8;

int TSize=sizeof(unsigned long long)\*8;

unsigned long long res=0;

for (i=0; i < TSize; i++)

if(data[i] == 1)

res+=pow(2.0, i);

return res;

}

byte \*ByteToBit(byte data)

{

int i;

byte \*res=new byte[8];

for (i=0; i < 8; i++)

{

if ((data >> i & 1) == 1) res[i]=1;

else res[i]=0;

}

return res;

}

byte BitToByte(byte \*data)

{

int i;

byte res=0;

for (i=0; i < 8; i++)

if(data[i] == 1)

res+=pow(2.0, i);

return res;

}

BMPheader ReadHeader(FILE \*Input)

{

BMPheader Img;

fread( &Img.bType, 1, sizeof(short), Input);

fread( &Img.bSize, 1, sizeof(int), Input);

fread( &Img.bReserved, 1, sizeof(int), Input);

fread( &Img.bOffset, 1, sizeof(int), Input);

fread( &Img.iSize, 1, sizeof(int), Input);

fread( &Img.bWidth, 1, sizeof(int), Input);

fread( &Img.bHeight, 1, sizeof(int), Input);

fread( &Img.bPlans, 1, sizeof(short), Input);

fread( &Img.BitPixel, 1, sizeof(short), Input);

fread( &Img.bCompress, 1, sizeof(int), Input);

fread( &Img.SizeCoImage, 1, sizeof(int), Input);

fread( &Img.bXPixel, 1, sizeof(int), Input);

fread( &Img.bYPixel, 1, sizeof(int), Input);

fread( &Img.bClrUsed, 1, sizeof(int), Input);

fread( &Img.bClrImportant, 1, sizeof(int), Input);

if(Img.bOffset == 54) Img.bPalitra=0;

else Img.bPalitra=1;

return Img;

}

byte \*ReadBMP(FILE \*Input, int \*count)

{

int i;

short c;

BMPheader Img;

byte \*GetPixel;

Img=ReadHeader(Input);

i=Img.bHeight\*Img.bWidth\*3;

GetPixel=new byte[i];

\*count=i;

for(i; i!=0; i--)

fread(&GetPixel[i-1], 1, sizeof(byte), Input);

return GetPixel;

}

void GetInfo(byte \*GetPixel, int p)

{

FILE \*Input=NULL;

int i=0;

int popravka=0;

int f;

int k=0;

unsigned char \*ImgDump, \*TxtDump, str;

unsigned long long DSize, TPos;

//TPos=sizeof(unsigned long long);

TPos=sizeof(unsigned long long)\*8;

TxtDump=new byte[8];

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

{

ImgDump=ByteToBit(GetPixel[i]);

if(k+p>7) popravka=k+p-8;

for(f=p-popravka; f>0; f--)

{

TxtDump[k]=ImgDump[f];

k++;

}

k--;

popravka=0;

free(ImgDump);

i++;

}

Input=fopen("Output.txt", "wb+");

free(TxtDump);

TxtDump=new byte[TPos];

for(k=0;k<TPos;k++)

{

ImgDump=ByteToBit(GetPixel[i]);

if(k+p>TPos-1)

popravka=k+p-TPos;

for(f=p-popravka; f>0; f--)

{

TxtDump[k]=ImgDump[f];

k++;

}

k--;

popravka=0;

free(ImgDump);

i++;

}

DSize=BitToLong(TxtDump);

free(TxtDump);

TxtDump=new byte[8];

for(TPos=0; TPos!=DSize; TPos++)

{

for(k=0; k!=8; k++)

{

ImgDump=ByteToBit(GetPixel[i]);

if(k+p>7) popravka=k+p-8;

for(f=p-popravka; f>0; f--)

{

TxtDump[k]=ImgDump[f];

k++;

}

k--;

popravka=0;

free(ImgDump);

i++;

}

str=BitToByte(TxtDump);

fwrite(&str,1,1,Input);

}

free(TxtDump);

if(Input!=NULL)

fclose(Input);

}

int main()

{

int count,i,n;

unsigned long long FSize;

byte \*TextFile;

byte \*GetPixel;

FILE \*Input=fopen("ttn.bmp", "rb+");

if( !Input ) return NULL;

printf("Stepen' ot 1 do 8\n");

scanf("%d",&n);

GetPixel=ReadBMP(Input, &count);

GetInfo(GetPixel,n);

fclose(Input);

}

**Результаты влияния степеней упаковки на полученное изображение**

6) Замена 5 битов



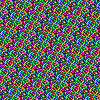
7) Замена 6 битов



8) Замена 7 битов



9) Замена 8 битов



1. Исходное изображение  
   
2. Замена 1 бита



1. Замена 2 бит   
   
2. Замена 3 бит  
   
3. Замена 4 бит



При любом изменении изображения информация становится недоступной.

**Ответы на контрольные вопросы:**

1) Стегосистема состоит из следующих основных элементов:

* прекодер – устройство, предназначенное для преобразования скрываемого сообщения к

виду, удобному для встраивания в сигнал-контейнер. (Контейнером называется информационная последовательность, в которой скрывают сообщение);

* стегокодер – устройство, предназначенное для осуществления вложения скрытого

сообщения в другие данные с учетом их модели;

* устройство выделения встроенного сообщения;
* стегодетектор – устройство, предназначенное для определения наличия

стегосообщения;

* декодер – устройство, восстанавливающее скрытое сообщение.

2) Основные виды атак на стегосистемы:  
Атаки против встроенного сообщения - направлены на удаление или порчу ЦВЗ путем манипулирования стего. Входящие в эту категорию методы атак не пытаются оценить и выделить водяной знак. Примерами таких атак могут являться линейная фильтрация, сжатие изображений, добавление шума, выравнивание гистограммы, изменение контрастности и т.д.

2. Атаки против стегодетектора – направлены на то, чтобы затруднить или сделать невозможной правильную работу детектора. При этом водяной знак в изображении остается, но теряется возможность его приема. В эту категорию входят такие атаки, как аффинные преобразования (то есть масштабирование, сдвиги, повороты), усечение изображения, перестановка пикселов и т.д.

2. Атаки против протокола использования ЦВЗ – в основном связаны с созданием ложных ЦВЗ, ложных стего, инверсией ЦВЗ, добавлением нескольких ЦВЗ.

4. Атаки против самого ЦВЗ – направлены на оценивание и извлечение ЦВЗ  из стегосообщения, по возможности без искажения контейнера. В эту группу входят такие атаки, как атаки сговора, статистического усреднения, методы очистки сигналов от шумов, некоторые виды нелинейной фильтрации  и другие.

3) Методы противодействия стеганографическим атакам

Первый способ заключается в построении необратимого алгоритма ЦВЗ. ЦВЗ должен быть адаптивным к сигналу и встраиваться при помощи однонаправленной функции, например, хэш-функции. Хэш-функция преобразует 1000 бит исходного изображения  в битовую последовательность , . Далее, в зависимости от значения  используется две функции встраивания ЦВЗ. Если , то используется функция , если , то функция , где - -й коэффициент изображения, - -й бит встраиваемого сообщения.

Второй способ решения проблемы прав собственности заключается во встраивании в ЦВЗ некоторой временной отметки, предоставляемой третьей, доверенной стороной. В случае возникновения конфликта лицо, имеющее на изображении более раннюю временную отметку, считается настоящим собственником.

Для защиты от атак типа аффинного преобразования можно использовать дополнительный (опорный) ЦВЗ. Этот ЦВЗ не несет в себе информации, но используется для «регистрации» выполняемых нарушителем преобразований. В детекторе ЦВЗ имеется схема предыскажения, выполняющая обратное преобразование. Здесь имеется аналогия с используемыми в связи тестовыми последовательностями. Однако, в этом случае атака может быть направлена именно против опорного ЦВЗ. Другой альтернативой является вложение ЦВЗ в визуально значимые области изображения, которые не могут быть удалены из него без существенной его деградации.

1. Под стойкостью различных стегосистем понимается их способность скрывать от квалифицированного нарушителя факт скрытой передачи сообщений, способность противостоять попыткам нарушителя разрушить, исказить, удалить скрытно передаваемые сообщения, а также способность подтвердить или опровергнуть подлинность скрытно передаваемой информации.
2. При сжатии изображения вероятность потерять секретное сообщение становится очень большой.

**Вывод**

Мною были созданы две программы:  
Кодер, который вставлял в изображение информацию.

Декодер, который расшифровывал эту информацию.  
На основе полученных данных могу сказать, что встраивание скрытой информации в изображение, позволит надёжно сохранить её от злоумышленников. Для ещё большей сохранности изображения, рекомендую перед встраиванием зашифровать его, например XOR- овским алгоритмом.