1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации

**ЛАБОРАТОРНАЯ РАБОТА № 4**

1. «**Основы стеганографической защиты информации**»
2. по дисциплине «Основы информационной безопасности»
3. Выполнил
4. студент гр. №4851003/00002 Скрипко И.А.

<*подпись*>

1. Преподаватель
2. асс. преподавателя Панков И.Д.

<*подпись*>

1. Санкт-Петербург
2. 2021
3. **Цель работы**

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

1. **Постановка задачи**
2. Создать произвольный графический файл формата BMP, который будет использоваться в качестве стегоконтейнера.
3. Создать программу-кодер, реализующую стеганографическое сокрытие секретного текстового файла в созданный графический файл. В программе во входных данных должна указываться степень упаковки, после чего шифроваться сообщение методом шифрования младших бит байтов палитры.
4. Визуально сравнить полученные изображения с зашифрованным сообщением при разной степени упаковки.
5. Создать программу-декодер и извлечь зашифрованное сообщение.
6. Описать поведение программы, если размер текстового файла будет превышать объем стегоконтейнера.
7. С помощью графического редактора выполнить преобразования изображения с зашифрованным сообщением, после чего запустить программу-декодер и сравнить полученное сообщение с изначальным.
8. **Теоретические исследования**
   1. **Описание формата файла изображения, исходный размер файла изображения, размер содержательной части графического файла**

Структура формата bmp включает в себя шапку, состоящую из 54 байт, и графическое изображение. На рисунке 1 можно увидеть начало файла формата bmp в HEX-редакторе.

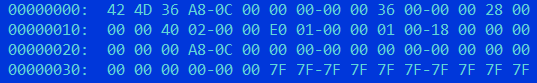


Рисунок 1 – Структура файла формата bmp

Первые 2 байта показываю сигнатуру формата, следующие 4 байта показывают размер файла. Далее 4 байта являются зарезервированной памятью (не используются). Далее 4 байта показывают смещение начала пиксельных данных. Затем 4 байта отведены на размер структуры заголовка, далее 4 байта на ширину изображения, затем 4 байта на высоту изображения, затем 2 байта на число плоскостей, 2 байта на количество бит на пиксель, затем 4 байта на метод сжатия, 4 байта на размер пиксельных данных. После чего идет само изображение.

Сведения об изображении приведены на рисунке 2.

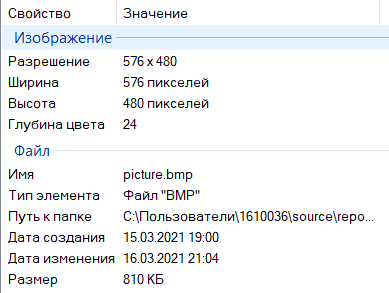


Рисунок 2 – Сведения об изображении

Максимальный объем стегоконтейнера, куда можно шифровать сообщение равен:

* 1. **Математический расчет максимальной длины скрываемого сообщения, помещаемого в созданный стегоконтейнер**

Пусть S – объем части стегоконтейнера, куда можно шифровать сообщение, в битах. Максимальная длина сообщения будет достигаться при замене 8/8 бит в байте каждого пикселя. Таким образом, максимальная длина сообщения, помещаемого в стегоконтейнер, может быть вычислена по следующей формуле:

, где 54 – размер «шапки»

* 1. **Элементы стегосистемы**

Предварительный кодер – устройство, предназначенное для преобразования скрываемого сообщения к виду, удобному для встраивания в стегоконтейнер.

Кодер – устройство, предназначенное для вложения сообщения в другие данные с учетом их модели.

Детектор – устройство, предназначенное для определения наличия стегосообщения.

Декодер – устройство, предназначенное для восстанавления скрытого сообщения.

* 1. **Понятие стойкости стегосистемы**

Стойкость стегосистемы – способность стегосистемы скрывать от квалифицированного нарушителя факт передачи сообщений, скрыти

* 1. **Методы противодействия стегонографическим атакам**

Для защиты можно использовать дополнительные ЦВЗ. Эти ЦВЗ не несут в себе информации, они фиксируют выполняемых нарушителем преобразований.

Другим методом защиты от подобных атак является блочный детектор. Модифицированное изображение разбивается на блоки небольших размеров, и для каждого блока анализируются все возможные искажения, для каждого изменения определяется коэффициент ЦВЗ. Преобразование, после которого коэффициент оказался наибольшим, считается реально выполненным нарушителем. Таким образом, появляется возможность обратить внесенные нарушителем искажения.

1. **Описание решения**
   1. **Кодер**

Пользователем вводятся имена файла формата bmp, стегоконтейнера, файла с секретным сообщением. Считывается размер секретного сообщения, выделяется память под сообщение. Считывается размер файла формата bmp, проверяется можно ли поместить сообщение в файл формата bmp. Далее сообщение представляется в двоичном виде в массиве типа char. Затем пользователем вводится степень упаковки, происходит проверка на корректность введенной степени упаковки. Затем в сообщение в двоичном представлении добавляются дополнительные нулевые биты для корректности шифрования. Далее копируется файл формата bmp в стегоконтейнер, после чего в неиспользуемые поля памяти (зарезервированная память – 7-10 байты) вставляется информация о секретном сообщении – степень упаковки и размер в битах. Затем происходит шифрование самого сообщения.

* 1. **Декодер**

Пользователем вводится имя стегоконтейнера и файла, куда будет сохранено расшифрованное сообщение. Затем из зарезервированной памяти считывается степень упаковки и размер сообщения, выделяется память под сообщение в битовом представлении и в приведенном в нормальный вид. Далее считываются пиксели и из них берутся младшие биты с сообщением, после чего они выставляются в правильном порядке, так как при шифровании они кодировались в обратном порядке по несколько бит (зависит от степени упаковки). Затем биты считывались по 8 штук в обратном порядке (так как при переводе в двоичное представление в программе-кодере младшие биты становились старшими). Далее расшифрованное сообщение выводится в указанный ранее пользователем файл.

1. **Тестирование и результаты работы программы**

Результаты работы программы приведены на рисунках 3-6.

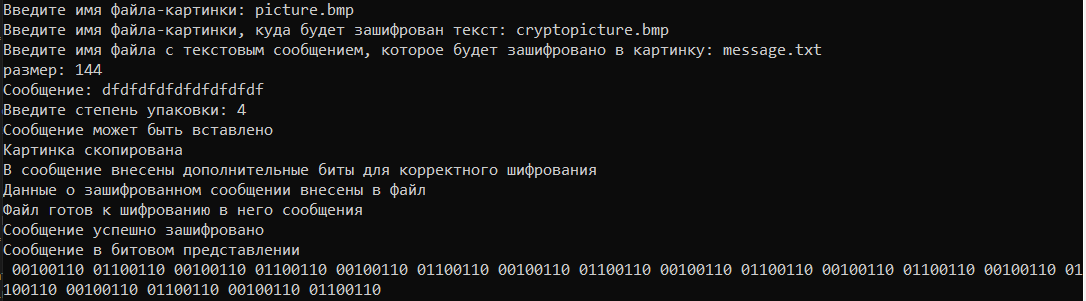


Рисунок 3 – Результат работы программы-кодера

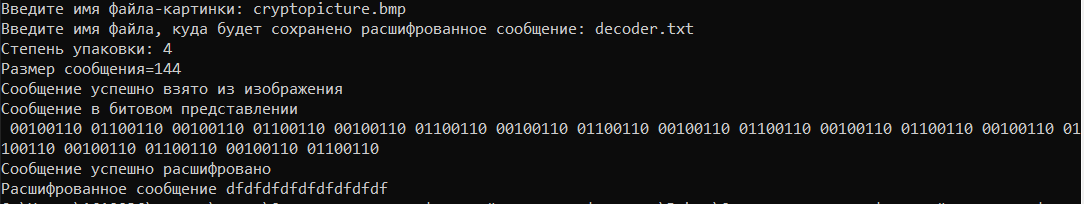


Рисунок 4 – Результат работы программы-декодера



Рисунок 5 – Стегоконтейнер при шифровании в него сообщения со степенью упаковки 8

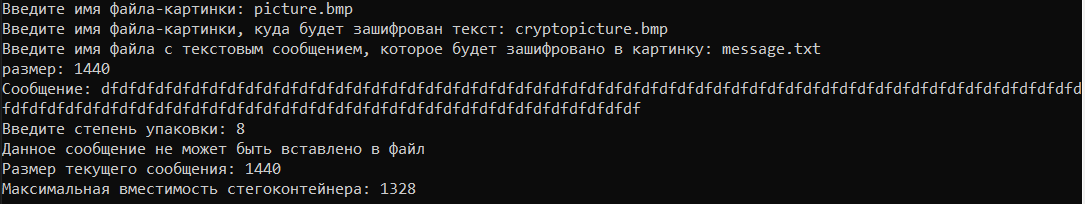


Рисунок 6 – Работа программы при попытки закодировать файл, размер которого превышает вместимость стегоконтейнера

1. **Выводы**

В ходе проделанной работы были приобретены навыки исследования свойств стегоконтейнеров, разработки стегосистем и их применения для сокрытия данных при передачи с помощью графических изображений. Разработанная стегосистема при кодировании 1 бит на байт хорошо скрывает сообщение, при кодировании 7-8 бит на байт видны изменения в изображении, что можно заменить на рисунке 6: в левом нижнем углу видно искажение. Также разработанная система не является стойкой к аффинным преобразованиям, так как теряется информация о зашифрованном сообщении.

**ПРИЛОЖЕНИЕ А**

Листинг программы «stega.c»

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "stega.h"

#include <stdio.h>

#include <Windows.h>

unsigned char\* result\_text;

int length\_of\_result\_text=0;

void binary\_text(int number)

{

int i = 0;

result\_text = (unsigned char\*)realloc(result\_text, length\_of\_result\_text + 8 \* sizeof(unsigned char));//добавляем памяти

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

{

result\_text[length\_of\_result\_text++] = (unsigned char)(number % 2);

number /= 2;

}

}

int coder()

{

int i=0,j=0;

char name\_of\_picture[SIZE\_OF\_NAME] = { 0 };

char name\_of\_text[SIZE\_OF\_NAME] = { 0 };

printf("Введите имя файла-картинки: ");

scanf("%s", name\_of\_picture);

FILE\* picture = fopen(name\_of\_picture, "r"); //ввод имени файла с картинкой и открытие файла для чтения

if (picture == NULL)

{

fclose(picture);

return FOPEN\_ERROR;

}

printf("Введите имя файла-картинки, куда будет зашифрован текст: ");

scanf("%s", name\_of\_picture);

FILE\* changed\_picture = fopen("cryptopicture.bmp", "r+"); //ввод имени файла картинки, куда будет зашифрован текст, и открытие файла для чтения и редактирования

if (changed\_picture == NULL)

{

fclose(picture);

fclose(changed\_picture);

return FOPEN\_ERROR;

}

printf("Введите имя файла с текстовым сообщением, которое будет зашифровано в картинку: ");

scanf("%s", name\_of\_text);

FILE\* text = fopen(name\_of\_text,"r");

if (text == NULL)

{ //ввод имени файла с текстовым сообщением, которое необходимо зашифровать

fclose(picture);

fclose(changed\_picture);

fclose(text);

return FOPEN\_ERROR;

}

fseek(text, 0, SEEK\_END);//устанавливаем позицию в конце файла

int lsize = ftell(text);//получаем размер в байтах

printf("размер: %d\n", lsize\*BIT\_IN\_BYTE);

unsigned char\* crypto = (unsigned char\*)malloc(lsize\*sizeof(unsigned char));//выделяем память

rewind(text);//ставим позицию в начало файла

fread(crypto, sizeof(unsigned char), lsize, text);//считываем из файла в массив символов

fseek(picture, 0, SEEK\_END);//устанавливаем позицию в конец файла

int picture\_size = ftell(picture);//получаем размер в байтах

printf("Сообщение: ");

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

printf("%c", crypto[i]); //вывод сообщения, которое будет зашифровано

printf("\n");

if (lsize == 0)

{

printf("Сообщение отсутствует");

return ZERO\_SIZE\_OF\_MESSAGE;

}

for (i = 0;i < lsize;i++) binary\_text((int)crypto[i]); //перевод сообщения в бинарный вид

printf("Введите степень упаковки: "); //сколько бит на байт будет шифроваться

int degree = 0;

scanf("%d", &degree);

if (degree == 0 || degree > 8)

{

printf("Степень упаковки не может принимать указанное вами значение\n"

"Размер текущего сообщения: %d\n"

"Максимальная вместимость стегоконтейнера: %d\n", length\_of\_result\_text,(picture\_size-54)\* BIT\_IN\_BYTE);//максимум можно 8 из 8 бит заменить

fclose(text);

fclose(picture);

fclose(changed\_picture);

fclose(text);

free(crypto);

return ERROR\_DEGREE;

}

/\*в файле .bmp сначала блок BITMAPFILEHEADER - идет формат(2 байта), затем размер файла(4 байта), затем зарезервированное поле(4 байта), которое не используется

затем смещение начальнаых пиксельных данных (4 байта)\*/

/\*затем идет блок BITMAPINFOHEADER - сначала идет размер структуры (4 байта), далее ширина(4 байта) и высота (4 байта) изображения

Далее число плоскостей (2 байта), далее количество бит на пиксель(2 байта), метод сжатия(4 байта), размер пиксельных данных (4 байта)

Разрешение пикселей на метр по ОХ(4 байта), разрешение пикселей на метр по ОY(4 байта), колиечство цветов в палитре(4 байта), количество важных цветов(4 байта), которое не используется\*/

if (length\_of\_result\_text % degree)//проверяем, сколько бит информации будет храниться в последнем пикселе

{

int extrasize = degree - length\_of\_result\_text % degree;

printf("Доп место: %d\n", extrasize);

result\_text = (unsigned char\*)realloc(result\_text, length\_of\_result\_text + extrasize \* sizeof(unsigned char));//дополняем сообщение, чтобы в последнем байте также было нужное количество бит

for (i = 0;i < extrasize;i++) result\_text[length\_of\_result\_text++] = (unsigned char)0;

}

unsigned char k='\0';

if ((picture\_size - BMP\_INFO) < length\_of\_result\_text/degree || length\_of\_result\_text >= BYTE\_RANGE \* BYTE\_RANGE)

{

printf("Данное сообщение не может быть вставлено в файл\n"

"Размер текущего сообщения: %d\n"

"Максимальная вместимость стегоконтейнера: %d\n", length\_of\_result\_text, (picture\_size-54) \* BIT\_IN\_BYTE);//максимум можно 8 из восьми бит заменить

fclose(text);

fclose(picture);

fclose(changed\_picture);

fclose(text);

free(crypto);

return ERROR\_SIZE\_OF\_MESSAGE;

}

printf("Сообщение может быть вставлено\n");

rewind(picture);//устанавливаем позицию в начало файла

for (i = 0;i < picture\_size;i++) //копирование картинки

{

fscanf(picture, "%c", &k);//берем из файла

fprintf(changed\_picture, "%c", k);//вставляем в новый файл

}

printf("Картинка скопирована\n");

printf("В сообщение внесены дополнительные биты для корректного шифрования\n");

fseek(changed\_picture, 6, SEEK\_SET);//устанавливаем курсор на неиспользуемое зарезервированное поле

fprintf(changed\_picture, "%c", degree);//сообщаем степень упаковки

fprintf(changed\_picture, "%c", length\_of\_result\_text / BYTE\_RANGE);//сообщаем размер, деленный на 256

fprintf(changed\_picture, "%c", length\_of\_result\_text % BYTE\_RANGE);//сообщаем остаток размера

printf("Данные о зашифрованном сообщении внесены в файл\n");

fseek(picture, BMP\_INFO, SEEK\_SET);//устанавливаем курсор на начало изображения

fseek(changed\_picture, BMP\_INFO, SEEK\_SET);//устанавливаем курсор на начало изображения

printf("Файл готов к шифрованию в него сообщения\n");

for (i = 0;i < length\_of\_result\_text;)

{

fscanf(picture, "%c", &k);//считываем пиксель

k >>= degree;//убираем младшие биты

for (j = 0;j < degree;j++)//возвращаем младшие биты со значения зашифрованного сообщения

{

k <<= 1;

k += result\_text[i++];

}

fprintf(changed\_picture, "%c", k);//вставляем измененный пиксель

}

printf("Сообщение успешно зашифровано\n");

fclose(text);

fclose(picture);

fclose(changed\_picture);

fclose(text);

free(crypto);

printf("Сообщение в битовом представлении\n");

for (i = 0;i < length\_of\_result\_text;i++) //вывод зашифрованного сообщения в бинарном виде

{

if (i % 8 == 0) printf(" ");

printf("%d", result\_text[i]);

}

return SUCCESS;

}

int decoder()

{

char name\_of\_picture[SIZE\_OF\_NAME] = { 0 };

char name\_of\_text[SIZE\_OF\_NAME] = { 0 };

printf("Введите имя файла-картинки: ");

scanf("%s", name\_of\_picture);

int memory1 = 0, memory2 = 0, memory=0, degree=0;

int i = 0, j = 0, z=0;

unsigned char k = '\0';

FILE\* changed\_picture = fopen(name\_of\_picture, "r"); //открытие файла картинки с зашифрованным сообщением

if (changed\_picture == NULL)

{

fclose(changed\_picture);

return FOPEN\_ERROR;

}

printf("Введите имя файла, куда будет сохранено расшифрованное сообщение: ");

scanf("%s", name\_of\_text);

FILE\* text = fopen(name\_of\_text, "w");

if (text == NULL) //открытие файла для сохранения сообщения

{

fclose(changed\_picture);

fclose(text);

return FOPEN\_ERROR;

}

fseek(changed\_picture, 6, SEEK\_SET);//устанавливаем курсор на неиспользуемое зарезервированное поле

fscanf(changed\_picture, "%c", &degree);//считываем степень упаковки

if (!degree)

{

fclose(changed\_picture);

fclose(text);

return ZERO\_SIZE\_OF\_MESSAGE;

}

printf("Степень упаковки: %d\n",degree);

fscanf(changed\_picture, "%c", &memory1);//считываем размер, деленный на 256

fscanf(changed\_picture, "%c", &memory2);//считываем остаток размера

memory = memory1 \* BYTE\_RANGE + memory2;//полный размер сообщения

printf("Размер сообщения=%d\n", memory);

unsigned char\* crypto = (unsigned char\*)malloc(memory \* sizeof(unsigned char));//выделяем память для побитового представления

unsigned char\* decode\_message = (unsigned char\*)malloc((memory \* sizeof(unsigned char))/ BIT\_IN\_BYTE);//выделяем память для сообщения в нормальном виде

memset(decode\_message, 0, (memory \* sizeof(unsigned char))/BIT\_IN\_BYTE);

memset(crypto, 0, memory \* sizeof(unsigned char));

fseek(changed\_picture, BMP\_INFO, SEEK\_SET);//устанавливаем курсор на начало изображения

for (i = 0;i < memory;)

{

fscanf(changed\_picture, "%c", &k);//считываем пиксель

for (j = 0;j < degree;j++)//считываем информацию

{

crypto[i++] = k % 2; //считываем младший бит и делаем сдвиг для считывания следующего младшего бита

k >>= 1;

}

}

for (i = 0;i < memory/degree;i++)//количество троек/четверок...

for (j = degree\*i,z=0;z <degree/2;z++)//ставим индекс на начало тройки/четверки...

{

unsigned char t = crypto[j+z]; //разворачиваем количество бит из каждого байта, так как шифрование происходило наоборот

crypto[j+z] = crypto[j+degree - z-1];//левая часть пары

crypto[j+ degree - z-1] = t;//правая часть пары

}

printf("Сообщение успешно взято из изображения\n");

printf("Сообщение в битовом представлении\n");

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

{ //вывод сообщения в битовом представлении

if (i % 8 == 0) printf(" ");

printf("%d", crypto[i]);

}

z = 0;

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

{

for (j = BIT\_IN\_BYTE-1;j >=0;j--)

{

decode\_message[z] += crypto[i\*8+j];

if (j!=0) decode\_message[z] <<= 1; //берется по 8 бит и составляется буква

}

z++;

}

printf("\nСообщение успешно расшифровано\n");

printf("Расшифрованное сообщение ");

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

{

printf("%c", decode\_message[i]); //вывод расшифрованного сообщения

fprintf(text, "%c", decode\_message[i]);

}

free(decode\_message);

free(crypto);

return SUCCESS;

}

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

printf("Выберете нужную программу:\n"

"1) Программа-кодер\n"

"2) Программа-декодер\n");

int choose = 0;

scanf("%d", &choose);

if (choose == 1)

{

int result = coder();

if (result != SUCCESS) return result;

}

else

{

int result = decoder();

if (result != SUCCESS) return result;

}

}

**ПРИЛОЖЕНИЕ B**

Листинг программы «stega.h»

#define SIZE\_OF\_NAME 255

#define FOPEN\_ERROR 254

#define SUCCESS 253

#define ERROR\_DEGREE 252

#define ZERO\_SIZE\_OF\_MESSAGE 251

#define ERROR\_SIZE\_OF\_MESSAGE 250

#define BMP\_INFO 54

#define BYTE\_RANGE 256

#define BIT\_IN\_BYTE 8