

Министерство образования Республики Беларусь  
Учреждение образования  
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей  
Кафедра информатики  
Дисциплина: Методы защиты информации

**ОТЧЕТ**  
к лабораторной работе №8  
на тему

«Программное средство сокрытия (извлечения) текстового сообщения в (из)  
JPEG изображение(я) на основе метода сокрытия в частотной области  
изображения»

Выполнил:

Д. С. Кончик

Проверил:

А. В. Герчик

Минск 2024

## СОДЕРЖАНИЕ

1 Постановка задачи.....	3
2 Краткие теоретические сведения.....	4
3 Результаты выполнения лабораторной работы.....	7
Выводы .....	8
Список использованных источников.....	9
Приложение А (обязательное) Листинг программного кода .....	10

## **1 ПОСТАНОВКА ЗАДАЧИ**

Целью выполнения данной лабораторной работы является реализация программного средства сокрытия и извлечения текстового сообщения из JPEG изображения на основе метода сокрытия в частной области изображения.

## 2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В отличие от криптографической защиты информации, предназначенной для сокрытия содержания информации, стеганографическая защита предназначена для сокрытия факта наличия (передачи) информации.

Методы и средства, с помощью которых можно скрыть факт наличия информации, изучает стеганография.

Методы и способы внедрения скрытой информации в электронные объекты относятся к компьютерной стеганографии.

Основными стеганографическими понятиями являются сообщение и контейнер.

Сообщением  $m \in M$  называют секретную информацию, наличие которой необходимо скрыть, где  $M$  - множество всех сообщений.

Контейнером  $b \in B$  называют несекретную информацию, которую используют для сокрытия сообщений, где  $B$  - множество всех контейнеров.

Пустой контейнер (контейнер-оригинал) – это контейнер  $b$ , не содержащий сообщения, заполненный контейнер (контейнер-результат) – это контейнер  $b$ , содержащий сообщение  $m$ .

### 2.1 Методы компьютерной стеганографии

Отметим, что несмотря на то, что методы тайнописи известны с древних времен, компьютерная стеганография является относительно новой областью науки. В настоящее время компьютерная стеганография находится на стадии развития.

Теоретическая база и методы стеганографии только формируются, нет общепризнанной классификации методов, не существуют критерии оценки надежности методов и механизмов стеганографических систем, производятся первые попытки проводить сравнительные характеристики методов.

Но уже сегодня специалисты признают, что «... на базе компьютерной стеганографии, являющейся одной из технологий информационной безопасности XXI века, возможна разработка новых, более эффективных нетрадиционных методов обеспечения информационной безопасности».

Анализ применяемых на практике методов компьютерной стеганографии позволяет выделить следующие основные классы:

1 Методы, основанные на наличии свободных участков в представлении/хранении данных.

2 Методы, основанные на избыточности представления/хранения данных.

3 Методы, основанные на применении специально разработанных форматов представления/хранения данных.

Подчеркнем, что методы внедрения скрытой информации в объекты зависят, прежде всего, от назначения и типа объекта, а также от формата, в котором представлены данные. То есть, для любого формата представления

компьютерных данных могут быть предложены собственные стеганографические методы.

Остановимся на стеганографических методах, которые часто применяются на практике.

Широко известен метод внедрения скрытой информации в младшие биты данных, представленных в цифровом виде. Метод основывается на том факте, что модификация младших, наименее значимых битов данных, представленных в цифровом виде, с точки зрения органов чувств человека не приводит к изменению функциональности и даже качества изображения или звука. Отметим, что информация, скрытая в последних битах цифрового контента, не является помехоустойчивой, то есть при искажениях или сжатии с потерей данных она теряется.

На практике используются также широкополосные сигналы и элементы теории шума. Информация скрывается путем фазовой модуляции информационного сигнала (несущей) с псевдослучайной последовательностью чисел. Используется и другой алгоритм: имеющийся диапазон частот делится на несколько каналов, и передача производится между этими каналами.

Достаточно развиты методы, применяемые для тайнописи в текстовых файлах:

1 Скрытые гарнитуры шрифтов. Данный метод основан на внесении малозаметных искажений, несущих смысловую нагрузку, в очертания букв.

2 Цветовые эффекты. Например, для символов скрываемого сообщения применяют белый цвет на белом фоне.

3 «Нулевой шифр». Этот метод основан на выборе определенных позиций символов (иногда используются известные смещения слов\\предложений\\ абзацев).

4 Обобщение акростиха. Метод заключается в том, что по определенному закону генерируется осмысленный текст, скрывающий некоторое сообщение.

5 Невидимые коды. Символы скрываемого сообщения кодируются определенным количеством дополнительных пробелов между словами или числом пустых строк.

Разработаны методы внедрения скрытой информации и для файлов в формате HTML:

1 В конец каждой строки добавляют определенное число пробелов, кодирующее скрываемую информацию;

2 Скрываемое сообщение размещают в специальном файле, у которого удаляют заголовок, причем такой заголовок хранится у получателя (скрываемое сообщение обычно дополнительно шифруется);

3 Присоединяют дополнительные страницы, на которых и размещают скрываемую информацию;

4 Записывают скрываемую информацию в мета-тэги (эти команды предназначены для сообщения информации о html-документе поисковым серверам и не видны при отображении страницы на экране);

5 Записывают скрываемую информацию в тэги с неизвестными программам-браузерам идентификаторами;

6 Применяют цветовые эффекты.

Особое внимание обратим на методы, применяемые для внедрения скрытой информации в исполняемые файлы.

Большинство из применяемых методов основано на наличии свободных участков в исполняемых файлах: полностью или частично свободные секторы (блоки) файла; структуры заголовков файлов в форматах EXE, NE-executable и PE-executable содержат зарезервированные поля; существуют пустоты между сегментами исполняемого кода и другие. Заметим, что именно такие методы компьютерной стеганографии традиционно используют авторы компьютерных вирусов для внедрения тел вирусов в исполняемые файлы. Обратим внимание, что для удаления скрытой таким образом информации нарушителю достаточно просто «обнулить» все имеющиеся свободные участки.

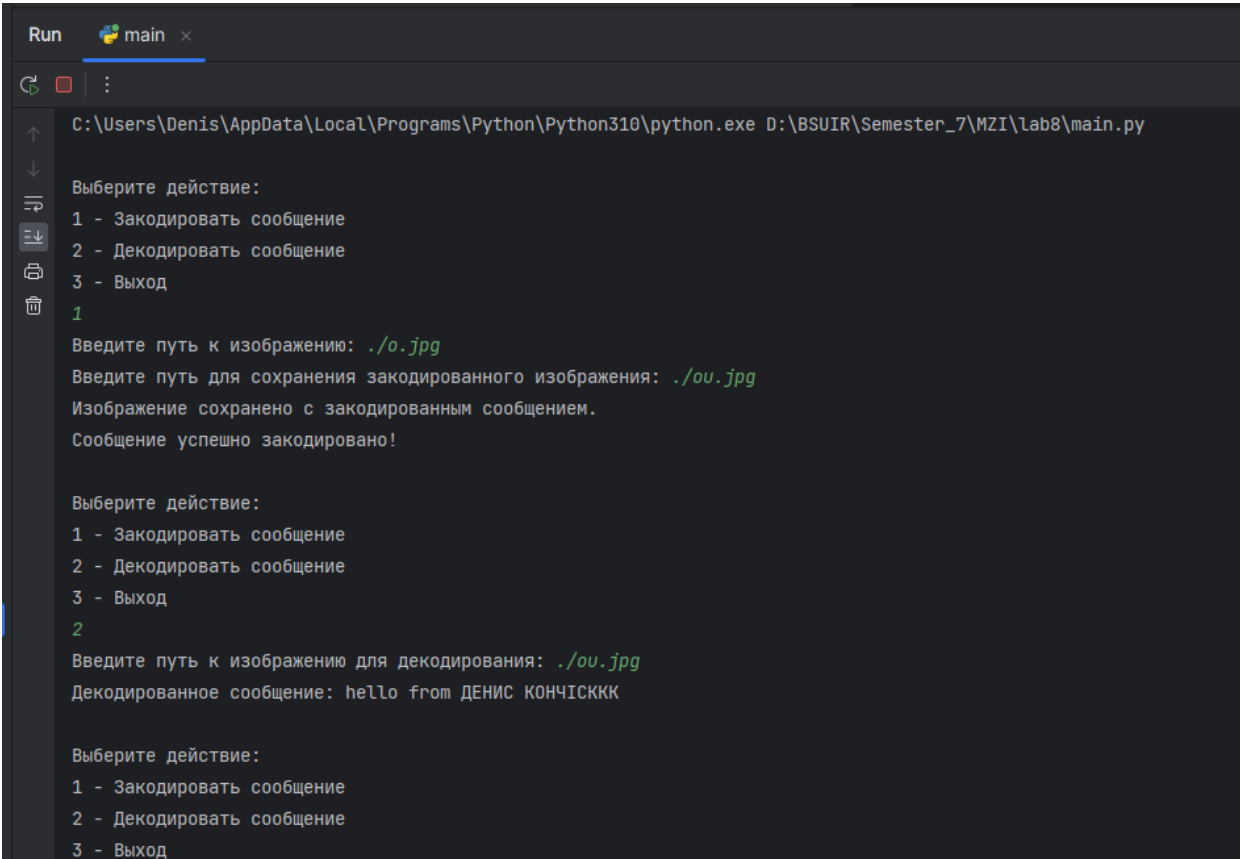
### 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В ходе выполнения лабораторной реализовано программное средство сокрытия и извлечения текстового сообщения из JPEG изображения на основе метода сокрытия в частной области изображения.

Сообщение, которое нам необходимо скрыть, находится в файле message.txt. Программа считывает необходимую информацию из файла, после чего обращается к картинке, в которой будет происходить сокрытие сообщения. С учетом того, что работа проводится с форматом JPEG существует необходимость знать и длину скрытого сообщения, чтобы при получении расшифрованного сообщения не выводить лишние символы, которые будут выводиться из-за дополнительного искажения с потерями.

Используя пиксели картинки каждый символ нашего сообщения будем хранить в младших байтах красных пикселей.

Результат выполнения лабораторной работы представлен на рисунке 3.1.



```
Run main x
C:\Users\Denis\AppData\Local\Programs\Python\Python310\python.exe D:\BSUIR\Semester_7\МЗИ\lab8\main.py

Выберите действие:
1 - Закодировать сообщение
2 - Декодировать сообщение
3 - Выход
1
Введите путь к изображению: ./o.jpg
Введите путь для сохранения закодированного изображения: ./ou.jpg
Изображение сохранено с закодированным сообщением.
Сообщение успешно закодировано!

Выберите действие:
1 - Закодировать сообщение
2 - Декодировать сообщение
3 - Выход
2
Введите путь к изображению для декодирования: ./ou.jpg
Декодированное сообщение: hello from ДЕНИС КОНЧИСККК

Выберите действие:
1 - Закодировать сообщение
2 - Декодировать сообщение
3 - Выход
```

Рисунок 3.1 – Результат выполнения лабораторной работы

## **ВЫВОДЫ**

В ходе данной лабораторной работы была разработана программное средство сокрытия и извлечения текстового сообщения из JPEG изображения на основе метода сокрытия в частной области изображения.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Методы компьютерной стеганографии [Электронный ресурс]. – Режим доступа: <https://www.vsavm.by/knigi/kniga3/1740.html/>. – Дата доступа: 06.11.2024.

[2] Стеганографические методы, устойчивые к jpeg сжатию [Электронный ресурс]. – Режим доступа: <https://cyberleninka.ru/article/n/steganograficheskie-metody-ustoychivye-k-jpeg-szhatuyu/>. – Дата доступа: 06.11.2024.

# ПРИЛОЖЕНИЕ А

## (обязательное)

### Листинг программного кода

#### Листинг 1 – Программный код файла main.py

```
from PIL import Image
import os

def set_bit(number, bit_position, bit_value):
    if bit_value == 1:
        return number | (1 << bit_position)
    else:
        return number & ~(1 << bit_position)

def get_message_bits(message):
    message_utf16 = message.encode("utf-16be") # Big-endian для однородной
    кодировки
    message_bits = []
    for byte in message_utf16:
        for i in range(7, -1, -1):
            message_bits.append((byte >> i) & 1)
    return message_bits

def encode_message(image_path, message, output_path):
    with Image.open(image_path) as img:
        bmp = img.convert("RGB")
        pixels = bmp.load()
        width, height = bmp.size

        # Проверка длины сообщения в битах
        message_bits = get_message_bits(message)
        if len(message_bits) > width * height:
            print("Сообщение слишком длинное для кодирования в этом
    изображении.")
            return

        # Сохраняем длину сообщения в байтах
        len_file_name =
f"len_{os.path.splitext(os.path.basename(output_path))[0]}.txt"
        with open(len_file_name, "w") as len_file:
            len_file.write(str(len(message_bits) // 8))

        bit_index = 0
        for y in range(height):
            for x in range(width):
                if bit_index >= len(message_bits):
                    break

                r, g, b = pixels[x, y]
                r = set_bit(r, 0, message_bits[bit_index])
                pixels[x, y] = (r, g, b)
                bit_index += 1

        bmp.save(output_path, format="PNG")
        print("Изображение сохранено с закодированным сообщением.")

def decode_message(image_path):
```

```

with Image.open(image_path) as bmp:
    bmp = bmp.convert("RGB")
    pixels = bmp.load()
    width, height = bmp.size

    len_file_name =
f"len_{os.path.splitext(os.path.basename(image_path))[0]}.txt"
    if not os.path.exists(len_file_name):
        print(f"Файл с длиной сообщения не найден: {len_file_name}")
        return ""

    with open(len_file_name, "r") as len_file:
        message_length = int(len_file.read()) # Длина в байтах

    # Рассчитываем общее количество бит
    total_bits = message_length * 8
    message_bytes = []
    byte_value = 0
    bit_index = 0

    for y in range(height):
        for x in range(width):
            if bit_index >= total_bits:
                break

            r, _, _ = pixels[x, y]
            bit = r & 1
            byte_value = (byte_value << 1) | bit
            bit_index += 1

            if bit_index % 8 == 0:
                message_bytes.append(byte_value)
                byte_value = 0

    decoded_message = bytes(message_bytes).decode("utf-16be")
    return decoded_message

def main():
    while True:
        print("\nВыберите действие:")
        print("1 - Закодировать сообщение")
        print("2 - Декодировать сообщение")
        print("3 - Выход")

        try:
            choice = int(input())
        except ValueError:
            print("Неверный ввод, пожалуйста, введите число от 1 до 3.")
            continue

        if choice == 1:
            input_path = input("Введите путь к изображению: ")
            if not os.path.exists("message.txt"):
                print("Файл 'message.txt' не найден!")
                continue
            with open("message.txt", "r", encoding="utf-8") as message_file:
                message = message_file.read()
            output_path = input("Введите путь для сохранения закодированного
изображения: ")
            encode_message(input_path, message, output_path)
            print("Сообщение успешно закодировано!")
        elif choice == 2:

```