Министерство образования Республики Беларусь
Белорусский государственный университет информатики и радиоэлектроники
Кафедра информатики

ОТЧЕТ
по лабораторной работе №8

Стеганографические методы

Выполнил:
студент гр. 653501
Шинкевич Г. С.

Проверил:
Артемьев В. С.

Минск 2019

ЗАДАНИЕ:

Реализовать программное средство сокрытия (извлечения) текстового сообщения в (из) изображение(я) на основе метода сокрытия в частотной области изображения.

Стеганография

Способ передачи или хранения информации с учётом сохранения в тайне самого факта такой передачи (хранения).

Цифровая стеганография — направление классической стеганографии, основанное на сокрытии или внедрении дополнительной информации в цифровые объекты, вызывая при этом некоторые искажения этих объектов. Как правило, данные объекты являются мультимедиа-объектами и внесение искажений, которые находятся ниже порога чувствительности среднестатистического человека, не приводит к заметным изменениям этих объектов.

Стегоалгоритмы встраивания информации в изображения

Скрытие данных в коэффициентах ДКП (дискретное косинусное преобразование)

Метод сокрытия данных, который заключается в изменении величин коэффициентов ДКП.

В данном способе изображение разделяется на блоки 8x8 пикселей, каждый из которых используется для шифрования одного бита сообщения. Шифрование начинается с произвольного подбора блока b_i используемого для шифрования i-го бита сообщения. Затем для подобранного блока b_i применяют ДКП: $B_i = D\{ b_i \}$. Для осуществления секретного канала абоненты должны выбрать два определенных коэффициента ДКП, которые будут применяться для шифрования секретной информации, обозначим их как (u1, v1) и (u2, v2). Данные коэффициенты — косинус-функции, соответствующие средним частотам. Такое соответствие позволит сохранить данные в необходимых областях сигнала при JPEG-сжатии, так как эти области не будут удаляться.

В случае выполнения неравенства B(u1, v1) > B (u2 , v2) считаем, что блок кодирует значение 1, в противном случае — 0. При внедрении данных выбранные коэффициенты обмениваются друг с

другом значениями, если их относительный размер не соответствует кодируемому биту. Следует отметить, что JPEG-сжатие влияет на относительные размеры коэффициентов, поэтому всегда должно выполняться условие $|B(u1, v1) - B(u2, v2)| > X$, где $X > 0$. Алгоритм становится устойчивее при увеличении X, но при этом теряется качество изображения. Обратное дискретное косинусное преобразование осуществляется после соответствующей корректировки коэффициентов. Получение зашифрованных данных осуществляется с помощью сравнения двух выбранных коэффициентов для каждого блока.

LSB

Суть этого метода заключается в замене последних значащих битов в изображении на биты скрываемого сообщения. Разница между пустым и заполненным контейнерами должна быть не ощутима для органов восприятия человека.

Пусть, имеется 24-х битное изображение в градациях серого. Пиксел кодируется 3 байтами, и в них расположены значения каналов RGB. Изменяя наименее значащий бит мы меняем значение байта на единицу. 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.

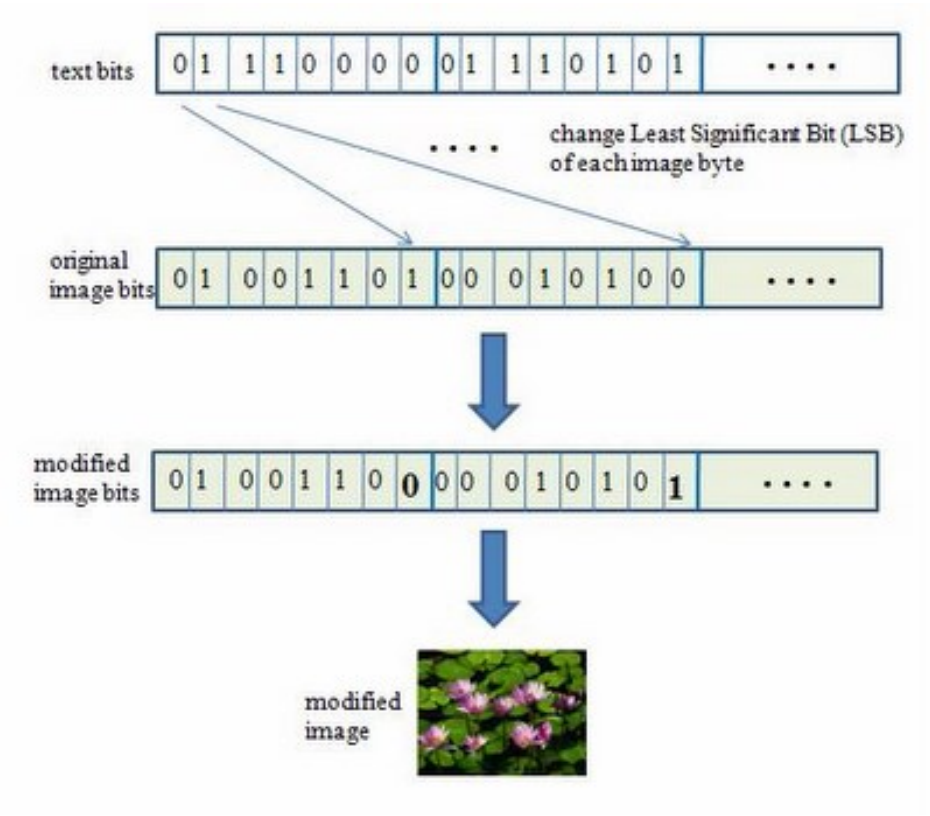

Рисунок 1. Метод LSB

ДЕМОНСТРАЦИЯ РАБОТЫ

Рисунок 2. Оригинальное изображение

Рисунок 3. Изображение с сообщением

```
Message: BSUIR
Extracted message: BSUIR
```

ВЫВОД

В результате лабораторной работы была написана программа для сокрытия тесктового сообщения в изображении и его извлечения.

ЛИСТИНГ ПРОГРАММЫ

```
from PIL import Image

WORD_LEN = 32


def bin_to_str(binary_string):
    text = ''
    for i in range(len(binary_string) // 8):
        bin_number = binary_string[i * 8:(i + 1) * 8]
        number = int(bin_number, 2)
        text += chr(number)
    return text


def bin_to_int(binary_string):
    return int(binary_string, 2)


def int_to_bin(number):
    return "{0:b}".format(number).zfill(WORD_LEN)


def str_to_bin(text):
    return ''.join(format(ord(char), 'b').zfill(8) for char in text)


def _set_bit(pixel_byte, data_bit):
    if data_bit:
        return pixel_byte | 1
    else:
        return pixel_byte & ~1


def _get_length_with_offset(img, width, height):
    data_length = ""
    for x in range(width):
        for y in range(height):
```

```
            if len(data_length) >= WORD_LEN:
                return bin_to_int(data_length[:WORD_LEN]), x, y

            pixel = list(img[x, y])
            for byte in pixel:
                data_length += str(byte & 1)


def embed(data, image):
    width, height = image.size
    img = image.load()
    color_depth = len(img[0, 0])

    padding = ""
    if WORD_LEN % color_depth != 0:
        padding = (color_depth - WORD_LEN % color_depth) * "0"

    data_len_bin = int_to_bin(len(data))

    data = data_len_bin + padding + data

    data = iter(data)
    for x in range(width):
        for y in range(height):
            pixel = list(img[x, y])
            for (i, pixel_byte), bit in zip(enumerate(pixel), data):
                if bit is None:
                    img[x, y] = tuple(pixel)
                    return

                pixel[i] = _set_bit(pixel_byte, int(bit))

            img[x, y] = tuple(pixel)


def extract(image):
    width, height = image.size
    img = image.load()
```

```
    data_length, offset_x, offset_y = _get_length_with_offset(img, width,
height)

    data = ""
    for x in range(offset_x, width):
        for y in range(offset_y, height):
            pixel = list(img[x, y])
            for byte in pixel:
                if len(data) == data_length:
                    return data

                data += str(byte & 1)


def embed_message(message, image):
    data = str_to_bin(message)
    embed(data, image)
    return image


def extract_message(image):
    return bin_to_str(extract(image))


if __name__ == '__main__':
    input_file = "image.png"
    output_file = "secret.png"
    message = "BSUIR"

    print(f"Message: {message}")
    input_image = Image.open(input_file)
    secret_image = embed_message(message, input_image)
    secret_image.save(output_file)

    output_image = Image.open(output_file)
    print(f"Extracted message: {extract_message(output_image)}")
```