Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

ОТЧЕТ

О ЛАБОРАТОРНОЙ РАБОТЕ

Лабораторная работа №2
по теме:
«Применение стеганографических методов для сокрытия информации»

Исполнитель, студент группы 201-361
_____ Н.А. Фельдбуш

Москва, 2023

Постановка задачи:

Напишите программу для внедрения в файл 28.bmp и извлечения из него хешкода файла leasing.txt, полученного с помощью алгоритма SHA-1. Метод реализации – LSB Replacement. Номера байтов-контейнеров должны содержаться в предварительно сгенерированном ключе. С помощью сторонних приложений оцените объемы получившегося и исходного изображений после сжатия.

Ход работы:

Выполним поставленную задачу при помощи средств языка программирования Python. Перед началом следует обратить внимание на две вспомогательных функции (рис.1).

```
33  #Получение данных о пикслеях в двоичном представлении.
34  def read_bmp_pixels_rgb(file_path):
35      img = Image.open(file_path)
36      pixel_data = list(img.getdata())
37      binary_pixel_data = [[format(pixel[0], '08b'), format(
38          pixel[1], '08b'), format(pixel[2], '08b')] for pixel in pixel_data]
39      return binary_pixel_data
40
41  #Создание файла по пикселям.
42  def create_image_from_pixels(pixel_data, width, height, file_path):
43      img = Image.new('RGB', (width, height))
44      pixels = img.load()
45      for y in range(height):
46          for x in range(width):
47              r, g, b = pixel_data[y * width + x]
48              pixels[x,  y] = (int(r, 2), int(g, 2), int(b, 2))
49      img.save(file_path)
50      return 1
51
```

Рисунок 1 – Вспомогательные функции

Функция “read_bmp_pixels_rgb”, отсекает заголовочные данные с помощью библиотеки “Pillow”, откуда вызвана функция “Image”, и передает только

значения байтов пикселей. Далее для каждого пикселя происходит перевод из байтов в биты, а после их запись в массив для дальнейшей работы.

Функция “create_image_from_pixels” создает новое изображение по полученным данным. С помощью библиотеки “Pillow” создается rgb файл, в который один за одним добавляются пиксели. Затем файл сохраняется по указанному пути.

Так же стоит обратить внимание на функции “keygen”, которая генерирует ключ, который в дальнейшем будет использоваться для замены битов (рис.2).

```python
#Создание ключа содержащего координаты и цвет пикселя с длиной равной длинне закодированного текстового файла.
def keygen(file_width, file_height, sha_len):
    a = []
    for i in range(len(sha_len)):
        rx = random.randrange(0, file_width*file_height)
        ry = random.randrange(0, 2)
        while [rx, ry] in a:
            print(rx)
            print(a)
            rx = random.randrange(0, file_width, file_height)
            ry = random.randrange(0, 2)
        else:
            a.append([rx, ry])
    with open('key.txt', 'w') as f:
        f.write(str(a))
    return(a)
```

Рисунок 2 – Создание ключа

В цикле for происходит выбор случайного пикселя(rx) и случайного байта цвета(ry). После происходит проверка, если этот байт уже был использован происходит перегенерации значений до тех пор, пока не будет найден уникальный байт. Затем происходит запись ключа в файл.

Замена битов происходит с помощью функции “LSB_Replacement” (рис.3).

```python
#Замена последнего бита в байте цвета.
def LSB_Replacement(file_width, file_height, sha, pixel_data, output_file_path):
    key = keygen(file_width, file_height, sha)
    # Ключ хранит в себе значения переменных pixel_data в которых хранятся хеш значения. Слева направа.
    for i in range(len(key)):
        pixel_data[key[i][0]][key[i][1]] = bin(
            (int(pixel_data[key[i][0]][key[i][1]], 2) & ~1) | int(sha[i]))[2:]  # Перевожу значения в инт чтобы было более понятно, что происходит.
    create_image_from_pixels(pixel_data, file_width,
                             file_height, output_file_path)

```

Рисунок 3 – Замена битов

В данной функции происходит получение сгенерированного ключа, который затем используем для получения нужного пикселя. После с помощью операции “& ~1” происходит обнуление последнего бита в последовательности бит пикселя. А затем с помощью операции “|” установка последнего бита в необходимое нам значение

Для получения хэш-кода из изображения необходимо использовать функцию “return_sha1_hash” (рис.4)

```
62  #Получение из файла изобржения зашифрованного хеша файла с помощью ключа
63  def return_sha1_hash(pixel_data):
64      fin = ""
65      with open("key.txt", "r") as f:
66          key = f.readline()
67      key = eval(key)
68
69      for k in range(len(key)):
70          a = pixel_data[key[k][0]][key[k][1]]
71          a = a[len(a)-1:]
72          fin += str(a)
73      return(fin)
74
```

Рисунок 4 – Замена битов

В функции происходит обратное действие: по ключу мы получаем значения, которые записывали в функции “LSB_Replacement”.

Результатом работы программы является новый файл изображения с закодированным внутри сообщением (рис.5).

Рисунок 5 – Замена битов