**Министерство науки и высшего образования РФ**

Федеральное государственное автономное

образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

факультет БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ

ТЕХНОЛОГИЙ

Основы стеганографии

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

«Встраивание информации в картинки»

Выполнил:

Студент группы N3351

Фам Хю Хоанг



Проверил: ассистент ФБИТ,

Университет ИТМО,

Давыдов Вадим Валерьевич

Санкт-Петербург

2020

**Цель работы:**

Целью данной лабораторной работы :

* Применение метода LSB для сокрытия
* Извлечение сообщения из стегоконтейнера
* Учет PSNR

**Теоретическая часть:**

LSB[1] -суть этого метода заключается в замене последних значащих битов в контейнере (изображения, аудио или видеозаписи) на биты скрываемого сообщения. Разница между пустым и заполненным контейнерами должна быть не ощутима для органов восприятия человека.

Методы LSB являются неустойчивыми ко всем видам атак и могут быть использованы только при отсутствии шума в канале передачи данных.

Обнаружение LSB-кодированного стего осуществляется по аномальным характеристикам распределения значений диапазона младших битов отсчётов цифрового сигнала.

Все методы LSB являются, как правило, аддитивными (А17 (Cox), L18D (Lange)).

BMP[2]-формат хранения растровых изображений, разработанный компанией Microsoft.В данном формате можно хранить только однослойные растры. На каждый пиксель в разных файлах может приходиться разное количество бит (глубина цвета). Microsoft предлагает битности 1, 2, 4, 8, 16, 24, 32, 48 и 64. В битностях 8 и ниже цвет указывается индексом из таблицы цветов (палитры), а при бо́льших — непосредственным значением. Цвет же в любом случае можно задать только в цветовой модели RGB (как при непосредственном указании в пикселе, так и в таблице цветов), но в битностях 16 и 32 можно получить Grayscale с глубиной до 16 и 32 бит, соответственно. Частичная прозрачность реализована альфа-каналом различных битностей, но при этом прозрачность без градаций можно косвенно получить RLE-кодированием.

PSNR[3]-Пиковое отношение сигнала к шуму обозначается аббревиатурой PSNR и является инженерным термином, означающим соотношение между максимумом возможного значения сигнала и мощностью шума, искажающего значения сигнала. Поскольку многие сигналы имеют широкий динамический диапазон, PSNR обычно измеряется в логарифмической шкале в децибелах.

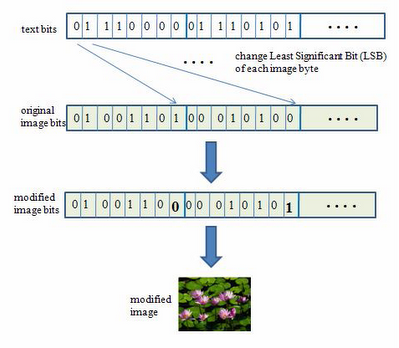
PSNR наиболее часто используется для измерения уровня искажений при сжатии изображений. Проще всего его определить через среднеквадратичную ошибку (СКО) или MSE

**Практическая часть:**

**Описание метода**

Суть метода заключается в следующем: Допустим, имеется 8-битное изображение в градациях серого. 00h обозначает чёрный цвет, FFh (11111111b) белый. Всего имеется 256 . Также предположим, что сообщение состоит из 1 байта например, 01101011b. При использовании 2 младших бит в описаниях пикселей, нам потребуется 4 пикселя. Допустим, они чёрного цвета. Тогда пиксели, содержащие скрытое сообщение, будут выглядеть следующим образом: 00000001 00000010 00000010 00000011. Тогда цвет пикселей изменится: первого — на 1/255, второго и третьего — на 2/255 и четвёртого — на 3/255. Такие градации, мало того, что незаметны для человека, могут вообще не отобразиться при использовании низкокачественных устройств вывода.

**Например сокрытие информации в изображении**



*Рисунок1.Пример использования НЗБ*

**Алгоритм встраивания секретного сообщения**

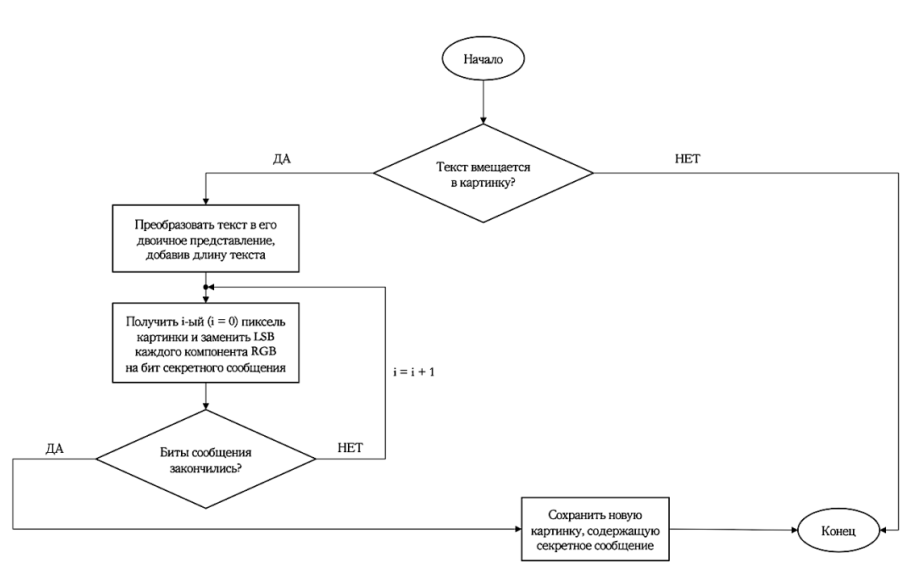
1.Загрузить копию картинки и текст секретного сообщения.Cообщение имеет необходимый размер, чтобы поместиться в изображение.

2. Преобразовать секретное сообщение в его двоичное представление, добавив длину фактического сообщения, чтобы на будущее знать, когда останавливаться при извлечении сообщения из картинки.

3. Перебрать каждый пиксель картинки:

* Разделить пиксель на его RGB компоненты;
* Заменить LSB каждого компонента битом сообщения;
* Остановить перебор, если биты сообщения закончились.

4. Сохранить новую картинку, содержащую секретное сообщение.



*Рисунок2. Блок-схема алгоритма встраивания секретного сообщения*

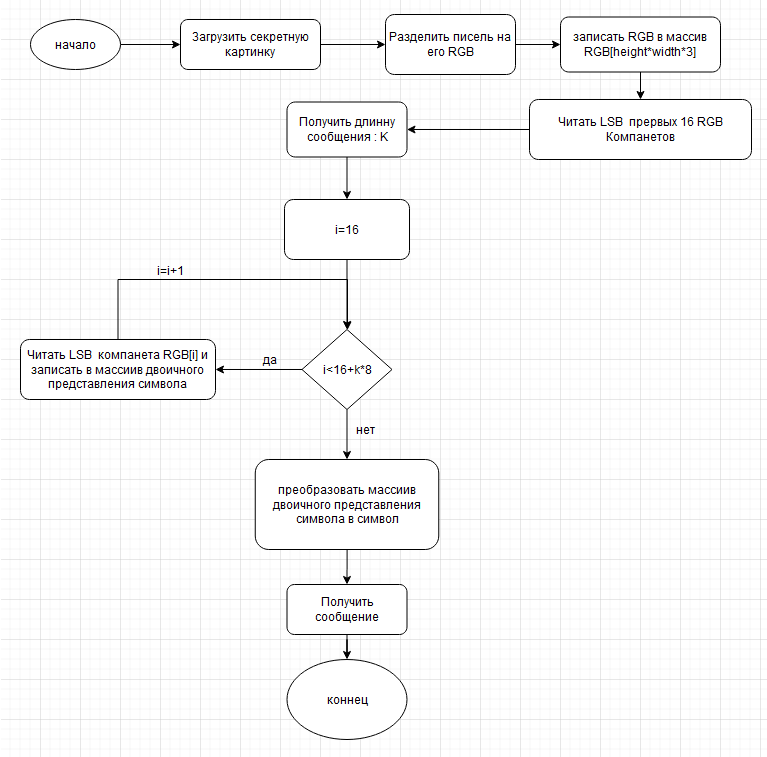
**Алгоритм извлечения секретного сообщения**

1.Загрузить секретную картинку

2. Перебрать каждый пиксель картинки:

* Разделить пиксель на его RGB компоненты и запиисать в массиив RGB (объем равно [height\*width\*3])
* Считать LSB первых 16 RGB компонентов и получить длину сообщения K
* Считать LSB из RGB[16] до RGB[16+K\*8] и записать в массив двоичного представления символа

4. Переобразовать его в символ и получить сообщение



*Рисунок 3. Блок-схема алгоритма извлечения секретного сообщения*

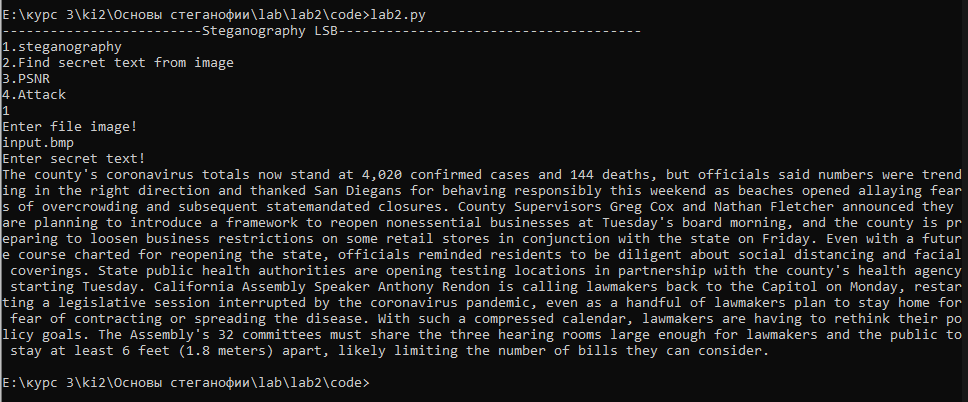
Программа была написана на языке Python. Программа делится на 4 модули:

1. Cтеганография
2. Пойск секретной информации
3. PSNR
4. Attack

Язык Python имеет множество библиотек, которые существенно упрощают разработку, поэтому некоторыеиз этих библиотек были использованы в данной работе:

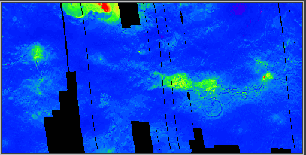
1. Pillow 7.1.1 (<https://pypi.org/project/Pillow/>)
2. Math (<https://pypi.org/project/maths/>)

**1.Steganography**



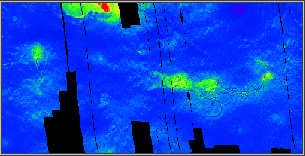
*Рисунок 4.* *Запуска программы*

**file input.bmp**



*Рисунок 5 file input.bmp*

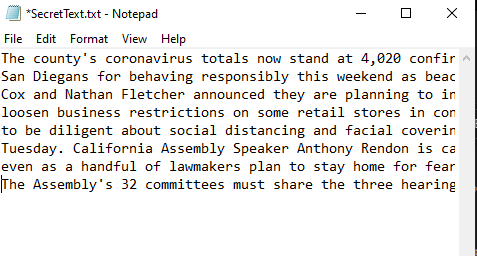
**File output.bmp**



*Рисунок 6 file output.bmp*

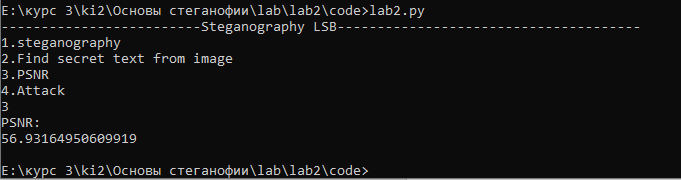
**2.Find secret text from image**

**File Secret text**



*Рисунок 7 file SecretText.txt*

**3.PSNR**

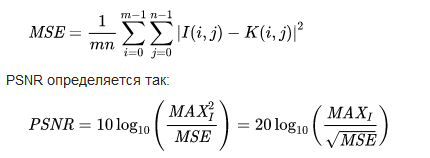


*Рисунок 8 Учет PSNR*

**Построить график PSNR :**

В заключении данного анализа построим график PSNR в зависимости количества встроенных слов в контейнер. Вспомним, что чем выше PSNR, тем меньше искажений присутствует в полученной картинке. Чтобы посчитать PSNR нужно для начала найти MSE (среднеквадратичная ошибка). Значение MSE, равное нулю, означает меньшее отклонение изображения от оригинала, в то время как значение MSE, превышающее 1, указывает на меньшее сходство (увеличивается по мере увеличения различий в цветах пикселей). Для PSNR значение приближенное к 100 означает, что в полученное изображение практически без искажений.

Рассмотрим формулы как считается PSNR и MSE, а затем построим график.



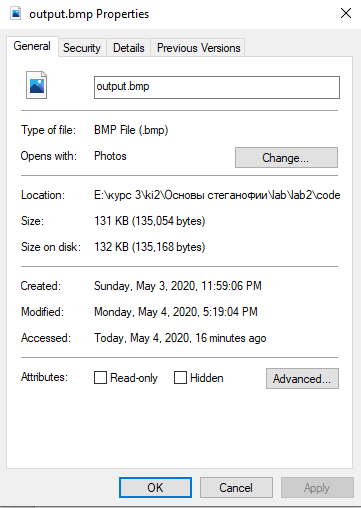
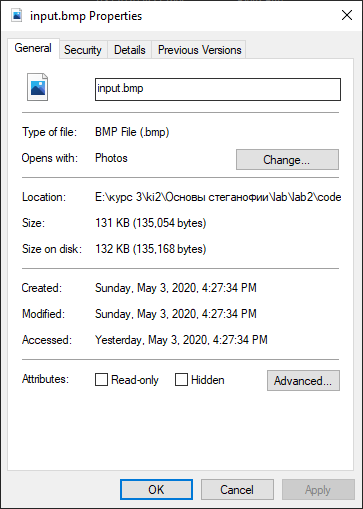
* I,K: сравниваемые картинки
* i,j: расположение пикселей
* m,n: ширина и высота сравниваемых картинок.
* MAXi-максимальное значение, принимаемое пикселем изображения.  
  Когда пиксели имеют разрядность 8 бит ,MAXi=255

|  |  |
| --- | --- |
| Слово | PSNR |
| 5 | 72.64896 |
| 10 | 69.80571 |
| 20 | 67.18881 |
| 30 | 65.59419 |
| 40 | 64.43865 |
| 50 | 63.49685 |

*Рисунок 9. График зависимости PSNR от количество встроенных слов*

Можно сделать вывод ,что при увеличении количества слова PSNR уменьшится

**Проведение экспертной оценки:**



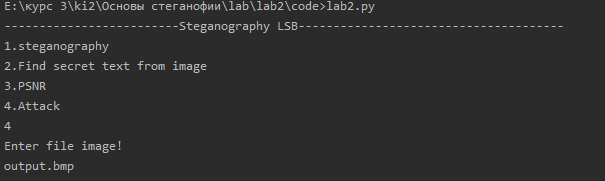
*Рисунок 10. Сравнение размера контейнера до и после встраивания*

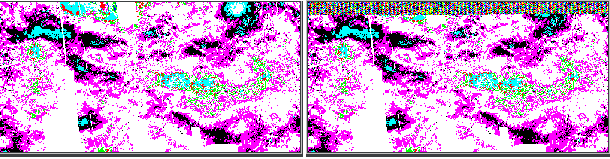
Можно сделать вывод, что объем картинки не изменяется. Итак, все требования безопасности соблюдены, и две картинки, в одной из которых десятая часть информации замещена произвольными данными, практически неразличимы.

**Атака на LSB-стеганографию**

Идея атаки заключается в том, чтобы обратить внимание на наименьший значащий бит каналов пикселей: если НЗБ равен нулевому, то его значение канала изменяется на 0, а если НЗБ равен единичному, то изменяется на 255. Например, пиксель, которая имеет (R,G,B) =(148, 165, 91) = (10010100, 10100101, 01011011), изменяется на RGB = (0, 255, 255).

Выполнять команду для атаки на 2 изображения и получать результат в рисунке





*Рисунок 11. Результаты атаки на 2 изображения до( на левом) и после(на правом)встраивания*

Можно легко определить разницу в 2 изображений, на первом изображении цветы случайно появляются, а на изображении с встроенной информацией появляется повторение по правилу, в соответствии с повторением кардов НЗБ. Эти повторении обусловлены повторении битовых кардов. Например, буквы a, b, c, d, … , o имеют общий кард “0110”, буквы p, q, r, s … , z имеют общий кард “0111” .

**Выводы:**

При выполнении данной лабораторной работы мною были изучены метод LSB. Я научился применять их и проводить последующую оценку их применению. По результатам работы были сделаны следующие выводы:

метод LSB подвержен статистическим атакам, что делает невозможным его применение на практике. Данный метод легко обнаружить, и он является наиболее известным стеганографическим алгоритмом. Тем не менее, это все еще эффективный способ сокрытия данных.

**Список использованной литературы:**

1. LSB[Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%B3%D0%B0%D0%BD%D0%BE%D0%B3%D1%80%D0%B0%D1%84%D0%B8%D1%8F#%D0%9C%D0%B5%D1%82%D0%BE%D0%B4\_LSB

2. BMP[Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/BMP

3. PSNR[Электронный ресурс] – URL: https://ru.wikipedia.org/wiki/%D0%9F%D0%B8%D0%BA%D0%BE%D0%B2%D0%BE%D0%B5\_%D0%BE%D1%82%D0%BD%D0%BE%D1%88%D0%B5%D0%BD%D0%B8%D0%B5\_%D1%81%D0%B8%D0%B3%D0%BD%D0%B0%D0%BB%D0%B0\_%D0%BA\_%D1%88%D1%83%D0%BC%D1%83

**Приложение**

from PIL import Image

import math

def ConvertStringToBinary(string):

    binary= ''

    for i in string:

        binary+=(format(ord(i), '08b'))

    return binary

def ConvertBinarryToString(binary):

    string = ""

    for i in range(0,len(binary),8):

        integer = int(binary[i:i+8], 2)

        character = chr(integer)

        string += character

    return string

def ConvertDecimalToBinary(number):

    binary=[]

    binray= "{0:016b}".format(number)

    return binray

def ConvertBinaryToDecimal(binary):

    dec=int(binary, 2)

    return dec

def ConvertDataToList(data,lenght):

    dataList=[]

    for i in range(0,lenght):

        dataList+=list(data[i])

    return dataList

def ModifyData(data,message):

    lenghtMessageBin=ConvertDecimalToBinary(len(message))

    for i in range(0,16):

        if(lenghtMessageBin[i]=='0'):

          if(data[i]%2==1):

            data[i]=data[i]-1

        else:

            if(data[i]%2==0):

                data[i]=data[i]+1

    messageBinary=ConvertStringToBinary(message)

    for i in range(0,len(messageBinary)):

        if(messageBinary[i]=='0'):

            if(data[i+16]%2==1):

                data[i+16]=data[i+16]-1

        else:

            if(data[i+16]%2==0):

                data[i+16]=data[i+16]+1

    return data

def ReadData(data):

    lengtMessageBinary=''

    for i in range(0,16):

        if(data[i]%2==0):

            lengtMessageBinary+='0'

        else:

            lengtMessageBinary+='1'

    lenghtMessage=ConvertBinaryToDecimal(lengtMessageBinary)

    message=''

    for i in range(16,16+lenghtMessage\*8):

        if(data[i]%2==0):

            message+='0'

        else:

            message+='1'

    return message

def Encryto(fileImage,message):

    image=Image.open(fileImage,'r')

    newImage=image.copy()

    width,height=newImage.size

    data=ConvertDataToList(list(newImage.getdata()),height\*width)

    data=ModifyData(data,message)

    dataImage=[]

    for j in range(0,height\*width\*3,3):

        dataImage.append(tuple(data[j:j+3]))

    newImage.putdata(dataImage)

    newImage.save('output.bmp')

    image.close()

    newImage.close()

def Decrypto(fileImage):

    image=Image.open(fileImage,'r')

    width,height =image.size

    data=ConvertDataToList(list(image.getdata()),height\*width)

    message=ConvertBinarryToString(ReadData(data))

    file\_secret = open('SecretText.txt',mode = 'w',encoding='UTF-8')

    file\_secret.write(message)

    file\_secret.close()

    image.close()

def PSNR(image1,image2):

    imageAfer=Image.open(image1,'r')

    imageBefor=Image.open(image2,'r')

    width,height=imageAfer.size

    dataAfter=ConvertDataToList(list(imageAfer.getdata()),height\*width)

    dataBefor=ConvertDataToList(list(imageBefor.getdata()),height\*width)

    sum=0

    for i in range(0,len(dataAfter)):

        sum+=math.pow((dataAfter[i]-dataBefor[i]),2)

    MSE=sum/(width\*height)

    PSNR=10\*math.log((255\*255/MSE),10)

    return PSNR

def Attack(fileImage):

    image=Image.open(fileImage,'r')

    newImage=image.copy()

    width,height=newImage.size

    data=ConvertDataToList(list(newImage.getdata()),height\*width)

    for i in range(0,len(data)):

        if(data[i]%2==0):

            data[i]=0

        else: data[i]=255

    dataImage=[]

    for j in range(0,height\*width\*3,3):

        dataImage.append(tuple(data[j:j+3]))

    newImage.putdata(dataImage)

    newImage.save('attack.bmp')

    image.close()

    newImage.close()

 ##---------------------------------------main programme-----------------------------------------------------------------

print('-------------------------Steganography LSB--------------------------------------')

print ('1.steganography'+'\n'+'2.Find secret text from image \n'+'3.PSNR\n'+'4.Attack')

option=int(input())

if(option==1):

     image=input('Enter file image!\n')

     message=input('Enter secret text!\n')

     Encryto(image,message)

elif( option==2):

     Decrypto(input('Enter file image!\n'))

     print ('Please Open file SecretText!')

elif(option==3):

    print ('PSNR:')

    print (PSNR('input.bmp','output.bmp'))

elif(option==4):

    Attack(input('Enter file image!\n'))