**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

Нгуен Туан Ань



Проверил: ассистент ФБИТ,

Университет ИТМО,

Давыдов Вадим Валерьевич

Санкт-Петербург

2020

**Цель работы**

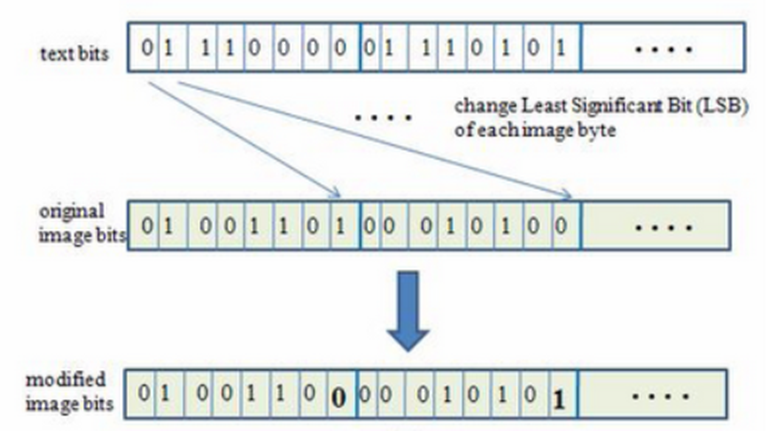
Реализовать модуль для встраивания информации в картинки и модуль для извлечения инфромации из картинок с помощью алгоритма НЗБ (наименьший значащий бит).

Построить график зависимости PSNR от объема встраивания, на основе полученого графика оценить качество изображения в зависимости от размера встраивания.

**Теоретическая часть**

**Метод замена наименее значащего бита (НЗБ)**

Суть метода замена наименее значащего бита (Least Significant Bits - LSB) заключается в сокрытии информации путем изменения последних битов изображения, кодирующих цвет на биты скрываемого сообщения. Разница между пустым и заполненным контейнерами должна быть не ощутима для органов восприятия человека.



*Рисунок 1. Пример использования НЗБ*

([1] <http://www.nestego.ru/2012/07/lsb.html>)

Метод наименее значащих битов (Least Significant Bit, LSB) наиболее распространен в электронной стеганографии. Он основан на ограниченных возможностях человеческих органов чувств, в силу чего люди не способны различать незначительные вариации цветов или звуков. Для простоты описания покажем принцип работы этого метода на примере 24-битного растрового RGB-изображения. Одна точка изображения в этом формате кодируется тремя байтами, каждый из которых отвечает за интенсивность одного из трех составляющих цветов.

([2] <https://ghostbasenji.blogspot.com/2018/08/steganography-method-LSB.html>)

**BMP формат**

BMP формат (от англ. Bitmap Picture) — формат хранения растровых изображений, разработанный компанией Microsoft. Файлы формата BMP могут иметь расширения .bmp, .dib и .rle.

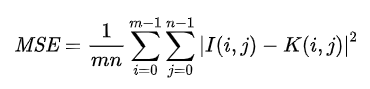
На каждый пиксель в разных файлах может приходиться разное количество бит (глубина цвета). Microsoft предлагает битности 1, 2, 4, 8, 16, 24, 32, 48 и 64. В битностях 8 и ниже цвет указывается индексом из таблицы цветов (палитры), а при больших -непосредственным значением. Цвет же в любом случае можно 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.

([3] <https://ru.wikipedia.org/wiki/BMP>)

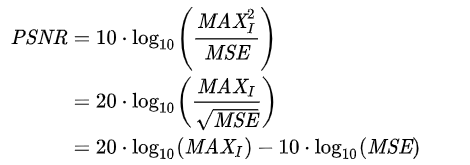
**Оценки качества изображения**

Отношение сигнал-шум Пик , часто сокращенно PSNR , является термином инжиниринг для соотношения между максимально возможной мощности в сигнале , и сила развращает шума , который влияет на точность его представления. Поскольку многие сигналы имеют очень широкий диапазон динамического , PSNR обычно выражается в терминах логарифмической децибел шкалы.

PSNR наиболее легко определяется с помощью средней квадратичной ошибки ( MSE ). Учитывая бесшумные m × n монохромного изображение I и его шумное приближение К , СКО определяются как:



PSNR (в дБ ) определяется так:



Здесь MAXI является максимально возможным значением пикселя изображения. Когда пиксели представлены с использованием 8 бит на выборку, это 255. В более общем плане , когда образцы представлены с использованием линейной ИКМ (Импульсно-кодовая модуляция) с B битами на выборку, MAXI 2В -1.

([4] <https://ru.qwe.wiki/wiki/Peak_signal-to-noise_ratio>)

**Практическая часть**

**Алгоритм встраивания секретного сообщения**

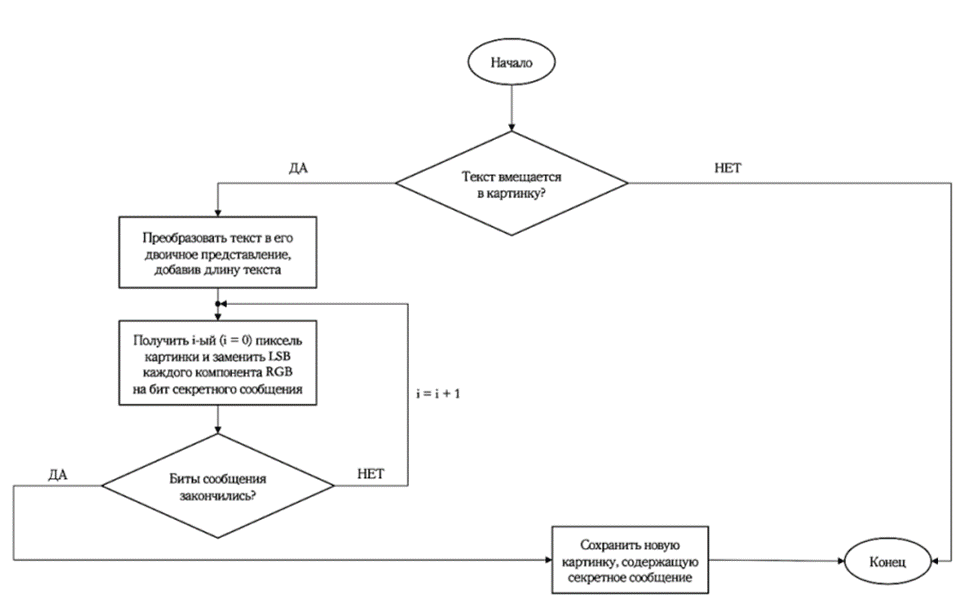
1. Загрузить копию картинки и текст секретного сообщения. Cообщение имеет необходимый размер, чтобы поместиться в картинку.

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.



*Рисунок 2. Блок-схема алгоритма встраивания секретного сообщения*

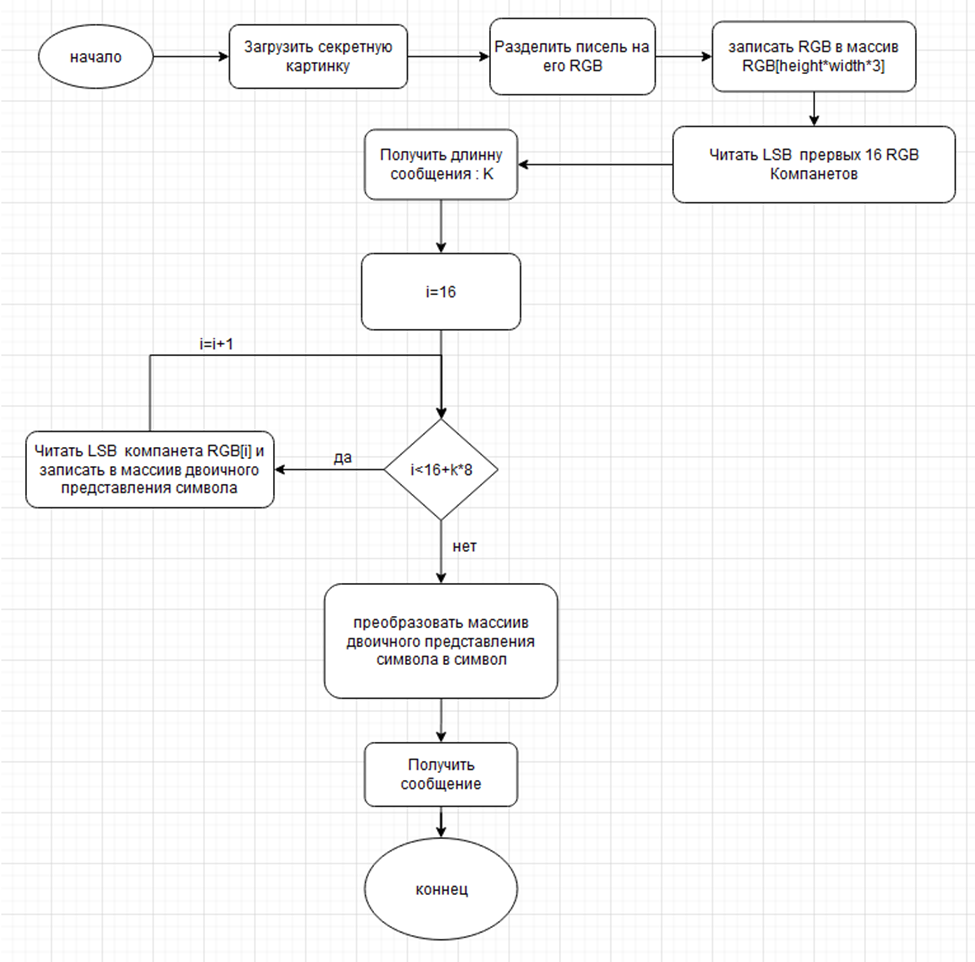
**Алгоритм извлечения секретного сообщения**

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.

3. Переобразовать его в символ и получить сообщение.



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

**Работа программы**

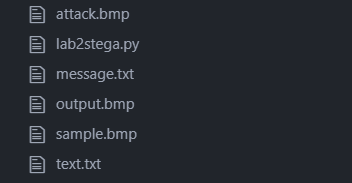
Для написания программ использовался язык Python 3 (скачать Python можно с официально сайта: <https://www.python.org/>), кодировался на Atom - бесплатный текстовый редактор с открытым исходным кодом для Linux, macOS, Windows с поддержкой плагинов (скачать с сайта: <https://atom.io/>).

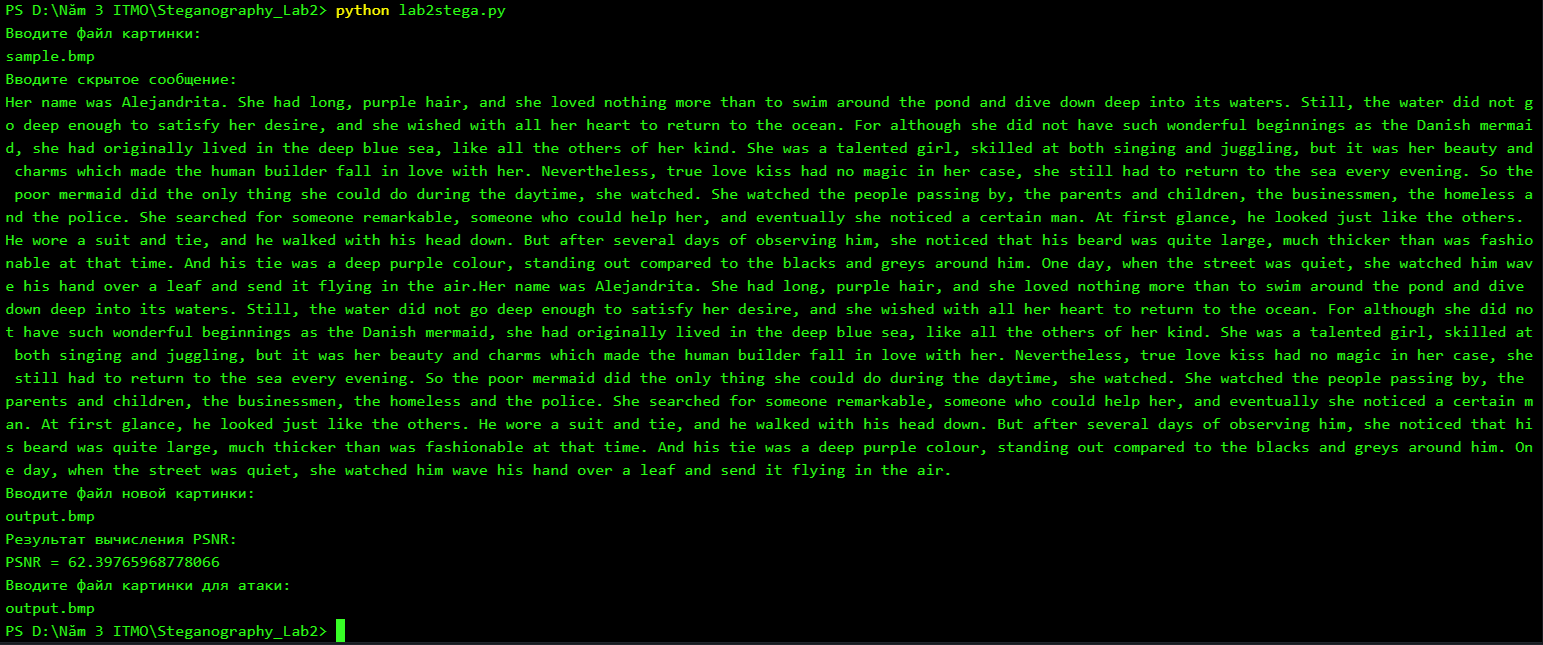
В данной работе были использованны следующие библиотеки:

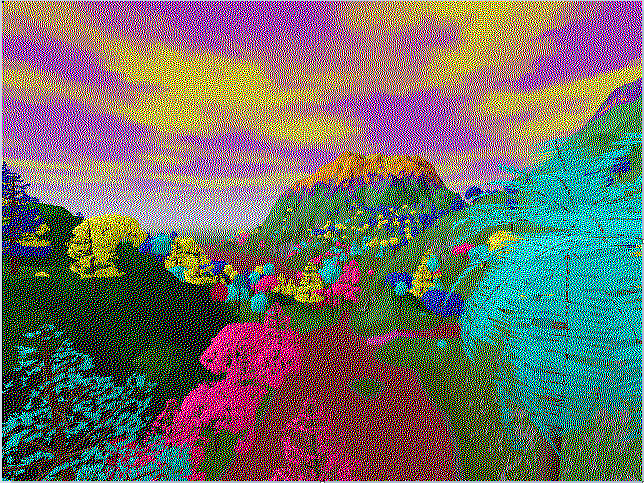
* Pillow 7.1.1 (<https://pypi.org/project/Pillow/>)
* Math (<https://pypi.org/project/maths/>)

В программе используются следующие файлы:

* Исходная картинка – sample.bmp
* Скрытое сообщение в файле message.txt
* Новая картинка – output.bmp
* После атаки – attack.bmp
* Скрытое сообщение из извлечения сохранится в файле text.txt



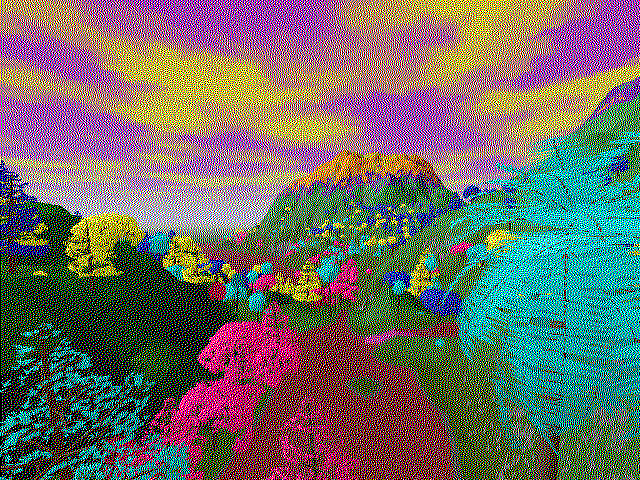
*Рисунок 4. Перечисление файлов*

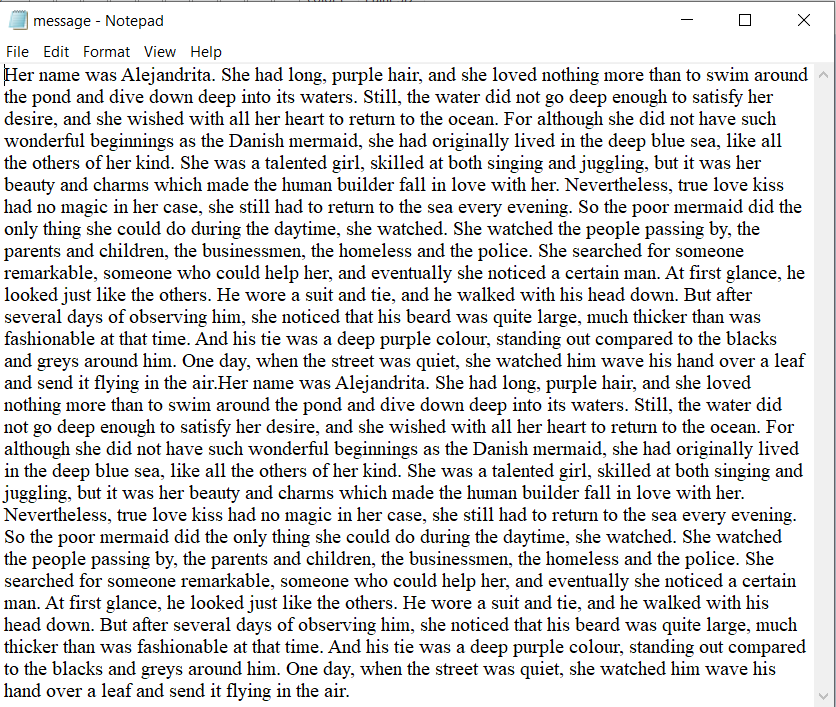
*Рисунок 5. Пример работы программы*

*Рисунок 6. Файл sample.bmp*

Изображенный контейнер (показан в файле sample.bmp) (можно скачать с сайта:

[5]<https://www.fileformat.info/format/bmp/sample/fa71e2b9ba0147f4b4b7dc19b50a4e4f/view.htm>).

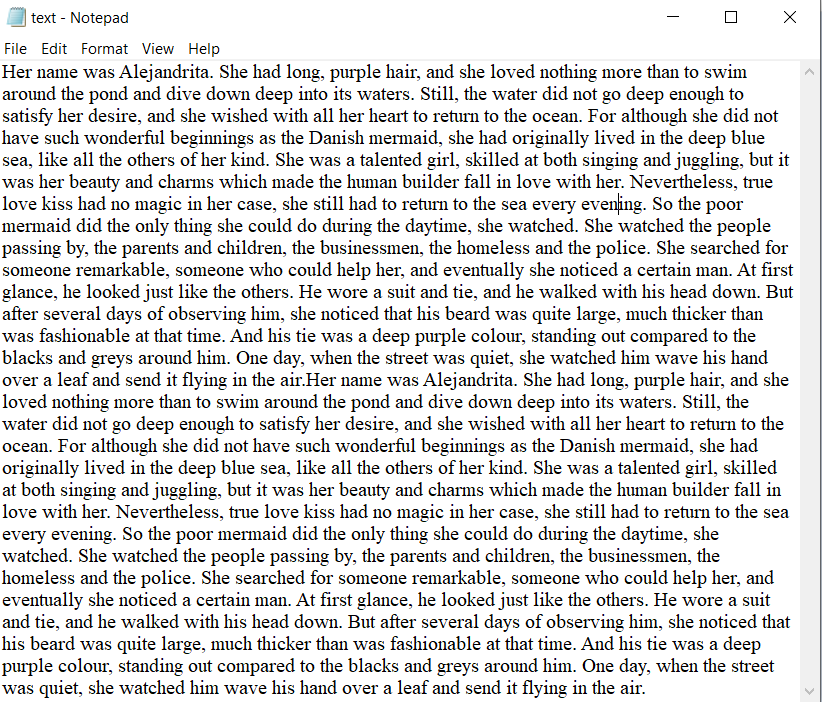


*Рисунок 7. Файл output.bmp*

*Рисунок 8. Файл message.txt*

Текстовое исходное сообщение - файл message.txt (информация из ссылки:

[6] <https://easystoriesinenglish.com/mermaid/>).



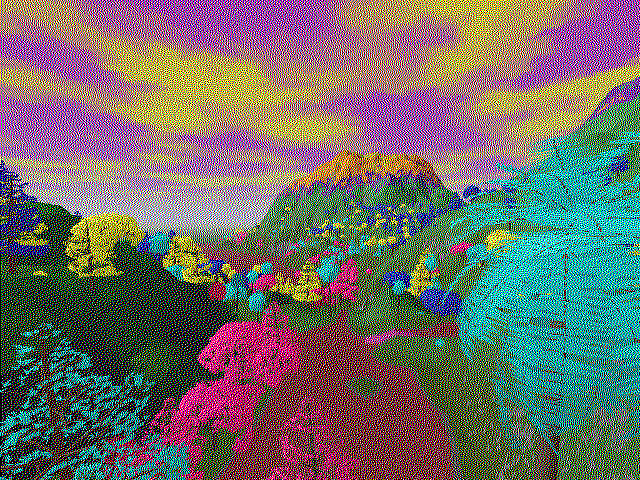
*Рисунок 9. Файл text.txt*

**Атака на LSB-стеганографию:**

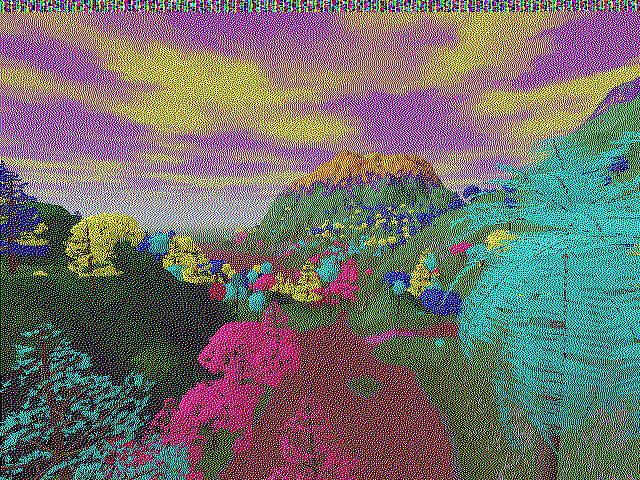
По методу «Наименьший значащий бит», нужно обратить внимание только на наименьший значащий бит. Замена значения пикселей выполнен по следующему принципу:

Если LSB компонента некоторой пиксели = 0, то значение такого компонента будет заменено на 0, а если LSB = 1, то значение = 255.

Пример результатов атаки на картике до встраивания и после встраивания информации.

****

*Рисунок 10. Результат атаки до встраивания информации*

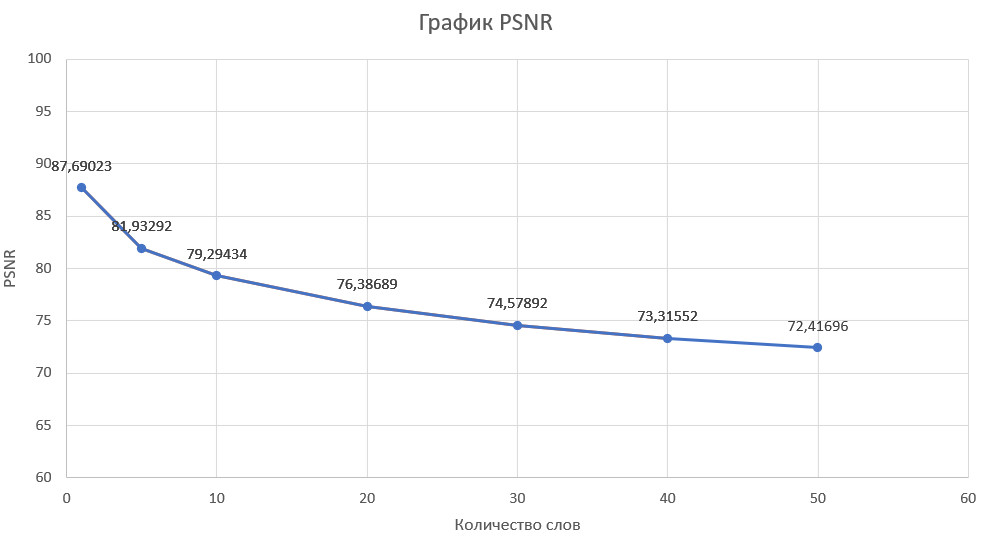
****

*Рисунок 11. Результат атаки**после встраивания**информации*

На рисунке 11 можно видеть необычное повторение в самой верхней рамке, в котором содержается встроенная информация. Это повторение связано с повторным появлением битовой кард «0110» и «0111» при преобразовании на двоичнуюформу с помощью таблицы ASCII (смотреть на сайте: [7] <https://www.ascii-code.com/>).

**Построение графика PSNR**

Встраивание по нарастающей: одно слово, пять, десять, 20, 30, 40, 50 и построение графика.



*Рисунок 12. График зависимости PSNR от количество встроенных слов*

Из графика можно делать вывод, что после встраивания кочество картинки хорошое (PSNR высокое) и если размера встраивания увеличиваетя то кочество изображения умешается.

**Оценка целесообразности**

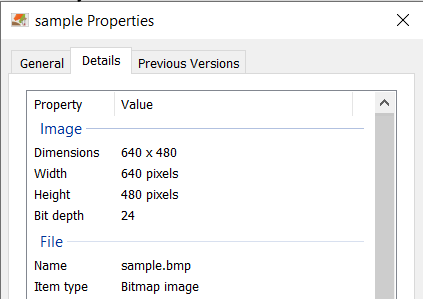
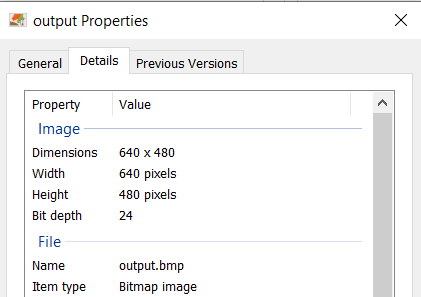
Данной метод целосообразуется в том, что:

* Кочество картинки после встраивания является хорошим (оценка на основе графика, показано на рисунке 12).
* Размер картинки до и после встраивания информации не изменяется, это показано на рисунке 13).



*Рисунок 13. Сравнение размера контейнера до и после встраивания*

* Максимальный размер информации, который можно встраивать в данной контейнере (901кБ –640х480 пик), достигается примерно 921000 битов, соответсвует примерно 115125 символов (если 1 символ - 8 бит).



*Рисунок 14. Сравнение размера контейнера (пиксель) до и после встраивания*

**Вывод**

При выполнении лабораторной работы были изучены основный метод LSB для встраивания информации в картинки (метод замены наименьший значащего бита). По результатам работы были сделаны следующие выводы:

* Качество скрытых цветных картинок трудно определить глазами.
* Простая атака сделана для определения, что ли существует встроенная информация в картинке.
* Для более безопасности и конфиденциальности информации после ее скрытия в картинке должно использовать интеллектуальный алгоритм шифрования секретное сообщение, прежде чем скрываться.

**Список использованной литературы**

1. Замена наименее значащего бита или LSB, 2012 г. [Электронный ресурс] – URL:

<http://www.nestego.ru/2012/07/lsb.html>.

1. Стеганография. Метод LSB, 2018 г. [Электронный ресурс] – URL:

<https://ghostbasenji.blogspot.com/2018/08/steganography-method-LSB.html>.

1. BMP [Электронный ресурс] – URL:

<https://ru.wikipedia.org/wiki/BMP>.

1. Пик отношение сигнал-шум - Peak signal-to-noise ratio [Электронный ресурс] – URL:

<https://ru.qwe.wiki/wiki/Peak_signal-to-noise_ratio>.

1. LAND3.BMP [Электронный ресурс] – URL:

<https://www.fileformat.info/format/bmp/sample/fa71e2b9ba0147f4b4b7dc19b50a4e4f/view.htm>.

1. The Other Mermaid [Электронный ресурс] – URL: <https://easystoriesinenglish.com/mermaid/>.
2. Таблицы ASCII [Электронный ресурс] – URL: <https://www.ascii-code.com/>.

**Приложение**

**Программа: lab2stega.py**

from PIL import Image

import math

def string\_to\_binary(text):

res = ''.join(format(ord(i), '08b') for i in text)

return res

def binary\_to\_string(binary):

str = ''

for i in range(0, len(binary), 8):

str += ''.join(chr(int(binary[i:i+8],2)))

return str

def decimal\_to\_binary(number):

res = []

res = "{0:016b}".format(number)

return res

def binary\_to\_decimal(binary):

num = int(binary,2)

return num

def data\_to\_list(data,lenght):

list\_data = []

for i in range(0,lenght):

list\_data += list(data[i])

return list\_data

def data\_after\_change(data, mes):

lenght\_mes\_bin = decimal\_to\_binary(len(mes))

for i in range(0,16):

if lenght\_mes\_bin[i] == '0':

if data[i]%2 == 1:

data[i] = data[i] - 1

else:

if data[i]%2 == 0:

data[i] = data[i] + 1

mes\_bin = string\_to\_binary(mes)

for i in range(0, len(mes\_bin)):

if mes\_bin[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 read\_data(data):

lenght\_mes\_bin = ''

for i in range(0,16):

if data[i]%2 == 0:

lenght\_mes\_bin += '0'

else:

lenght\_mes\_bin += '1'

lenght\_mes = binary\_to\_decimal(lenght\_mes\_bin)

mes = ''

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

if data[i]%2 == 0:

mes += '0'

else:

mes += '1'

return mes

def secret\_mes(file):

mess = open('message.txt','r').read()

return mess

def encrypt\_file(file,mes):

image = Image.open(file, 'r')

new\_image = image.copy()

width, height = new\_image.size

data = data\_to\_list(list(new\_image.getdata()),width\*height)

data = data\_after\_change(data,mes)

data\_of\_image = []

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

data\_of\_image.append(tuple(data[i:i+3]))

new\_image.putdata(data\_of\_image)

new\_image.save('output.bmp')

image.close()

new\_image.close()

def decrypt\_file(file):

image = Image.open(file, 'r')

width, height = image.size

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

mes = binary\_to\_string(read\_data(data))

file\_text = open('text.txt', 'w', encoding ='UTF-8')

file\_text.write(mes)

file\_text.close()

image.close()

def PSNR(image, new\_image):

image = Image.open(image,'r')

new\_image = Image.open(new\_image, 'r')

width, height = new\_image.size

data\_new = data\_to\_list(list(new\_image.getdata()),width\*height)

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

sum = 0

for i in range(0,len(data\_new)):

sum += math.pow((data\_new[i] - data[i]),2)

MSE = sum/(width\*height)

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

return PSNR

def attack\_image(file):

image = Image.open(file, 'r')

new\_image = image.copy()

width, height = new\_image.size

data = data\_to\_list(list(new\_image.getdata()),width\*height)

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

if data[i]%2 == 0:

data[i] = 0

else:

data[i] = 255

data\_of\_image = []

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

data\_of\_image.append(tuple(data[k:k+3]))

new\_image.putdata(data\_of\_image)

new\_image.save('attack.bmp')

image.close()

new\_image.close()

print("Вводите файл картинки: ")

image = input()

print("Вводите скрытое сообщение: ")

mes = secret\_mes('message.txt')

print(mes)

encrypt\_file(image,mes)

print("Вводите файл новой картинки:")

new\_image = input()

decrypt\_file(new\_image)

print("Результат вычисления PSNR:")

print(f"PSNR = {PSNR('sample.bmp','output.bmp')}")

print("Вводите файл картинки для атаки:")

file = input()

attack\_image(file)

**Файл: message.txt**

Her name was Alejandrita. She had long, purple hair, and she loved nothing more than to swim around the pond and dive down deep into its waters. Still, the water did not go deep enough to satisfy her desire, and she wished with all her heart to return to the ocean. For although she did not have such wonderful beginnings as the Danish mermaid, she had originally lived in the deep blue sea, like all the others of her kind. She was a talented girl, skilled at both singing and juggling, but it was her beauty and charms which made the human builder fall in love with her. Nevertheless, true love kiss had no magic in her case, she still had to return to the sea every evening. So the poor mermaid did the only thing she could do during the daytime, she watched. She watched the people passing by, the parents and children, the businessmen, the homeless and the police. She searched for someone remarkable, someone who could help her, and eventually she noticed a certain man. At first glance, he looked just like the others. He wore a suit and tie, and he walked with his head down. But after several days of observing him, she noticed that his beard was quite large, much thicker than was fashionable at that time. And his tie was a deep purple colour, standing out compared to the blacks and greys around him. One day, when the street was quiet, she watched him wave his hand over a leaf and send it flying in the air.Her name was Alejandrita. She had long, purple hair, and she loved nothing more than to swim around the pond and dive down deep into its waters. Still, the water did not go deep enough to satisfy her desire, and she wished with all her heart to return to the ocean. For although she did not have such wonderful beginnings as the Danish mermaid, she had originally lived in the deep blue sea, like all the others of her kind. She was a talented girl, skilled at both singing and juggling, but it was her beauty and charms which made the human builder fall in love with her. Nevertheless, true love kiss had no magic in her case, she still had to return to the sea every evening. So the poor mermaid did the only thing she could do during the daytime, she watched. She watched the people passing by, the parents and children, the businessmen, the homeless and the police. She searched for someone remarkable, someone who could help her, and eventually she noticed a certain man. At first glance, he looked just like the others. He wore a suit and tie, and he walked with his head down. But after several days of observing him, she noticed that his beard was quite large, much thicker than was fashionable at that time. And his tie was a deep purple colour, standing out compared to the blacks and greys around him. One day, when the street was quiet, she watched him wave his hand over a leaf and send it flying in the air.