**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»

Работу выполнил студент

группы N3352, Нгуен Тхай Хынг



Работу проверил: ассистент ФБИТ,

Университет ИТМО,

Давыдов Вадим Валерьевич

Санкт-Петербург

2020

**I. Цель работы**

Встраивание текстовой информации в картинки, используя метода LSB (наименьший значимый бит). Файл картики в формате .bmp, цвет представлен сочетанием трех цветовых компонент RGB.

Извлечение тектовой информации из стегоконтейнера.

Построение графика значений PSNR исходного изображения и изображения с информацией, встроенные по наростающей один символ, 5, 10, 20, 30, 40,50.

Проведение простого атака.

Оценка целесообразности метода с реальными примерами.

**II. Теория**

Наименьший значимый бит или НЗБ (Least Significant Bit - LSB) - это позиция бита в двоичном целом числе, определяющая, является ли число четным или нечетным, когда это двоичное число преобразуется в десятичное [1].

Метод наименьшего значимого бита (LSB) является одним из стеганографических методов, в котором наименьший значащий бит изображения заменяется битом данных [2].

В этом работе, используется метод замены наименьшего значащего бита с картинками в формате “.bmp”. Формат файла BMP, также известный как файл растрового изображения или формат файла растрового изображения, не зависящий от устройства, или просто растровое изображение - это формат файла растрового графического изображения, используемый для хранения растровых цифровых изображений независимо от устройства отображения, особенно в Microsoft Windows операционные системы [3].

RGB (Red, Green, Blue – красный, зеленый, синий) – аддитивная цветовая модель, описывающая способ кодирования цвета для цветовоспроизведения с помощью трех цветов, которые принято называть основными [4].

Для оценки качества изображения RGB (три канала в одной пиксели) используют меру среднеквадратического искажения (Mean Squared Error - MSE) [5].

MSE =

Где:

m,n – размер изображения.

k – количество каналов в одной писели.

I, K – значение каналов пикселей исходного и выходного изображения.

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 [5].

PSNR = 10log10()

Где:

MAXI - это максимальное значение, принимаемое пикселем изображения. Когда пиксели имеют разрядность 8 бит, MAXI = 255.

**III. Практика**

Для выполнения этой лабораторной работы, я написал 3 программы на языке Python 3.8 с помощью IDE Visual Studio Code. В программах я использовал библиотеку PIL для обработки файла изображения.

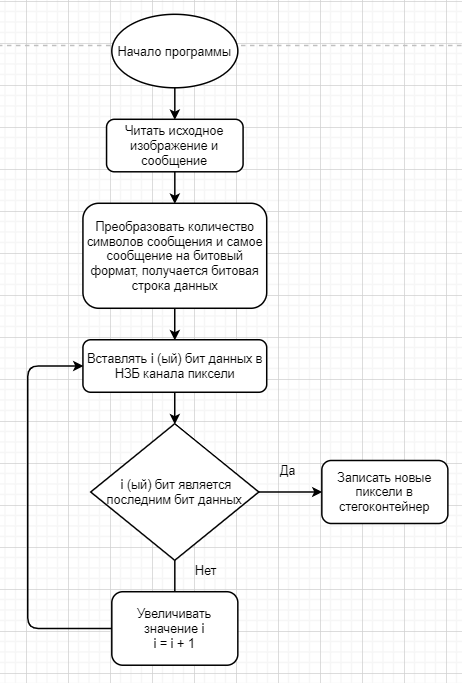
- Программа “image\_stegano.py” для встраивания и извлечения сообщения

- Программа “psnr.py” для расчета значений PSNR

- Программа “detect.py” для провеления простого атака

**Часть 1**:

**1. Блок-схема алгоритма встраивания**:



Рисунка 1. Блок-схема алгоритма встраивания

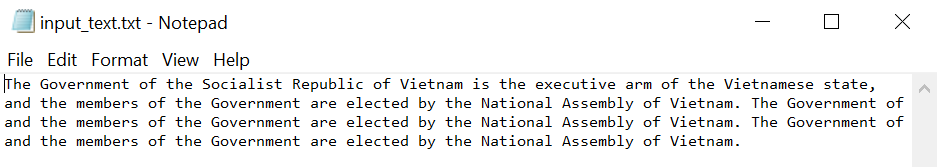
**2. Блок-схема алгоритма извлечения**:



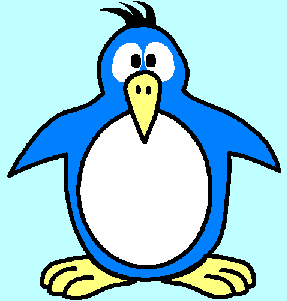
Рисунка 2. Блок-схема алгоритма извлечения

**3. Встраивание и извлечение**:

- Секретное сообщение, которое мы хотим встраивать в файле “input\_text.txt”:

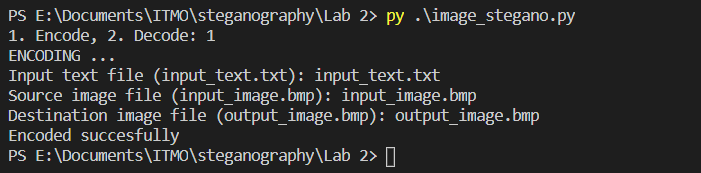


- Исходное изображение “input\_image.bmp”:

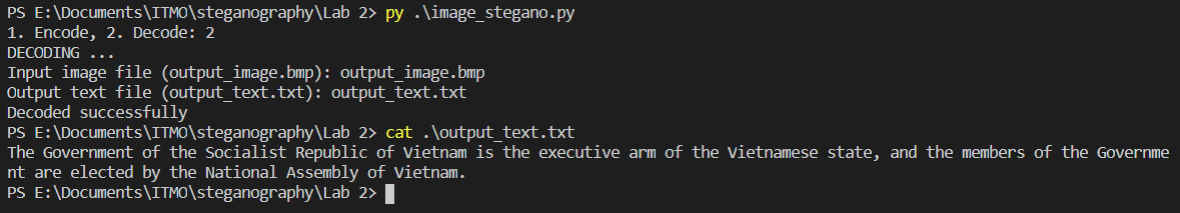


Рисунка 3. Исходное изображение

- Запуск программы “image\_stegano.py”:



Рисунка 4. Прогамма встроила сообщение успешно

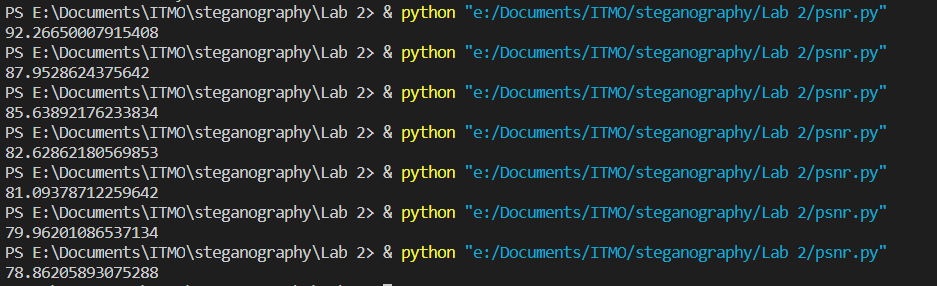


Рисунка 5. Прогамма извлекла сообщение успешно

**Часть 2**:

**1. Построение графика PSNR**:

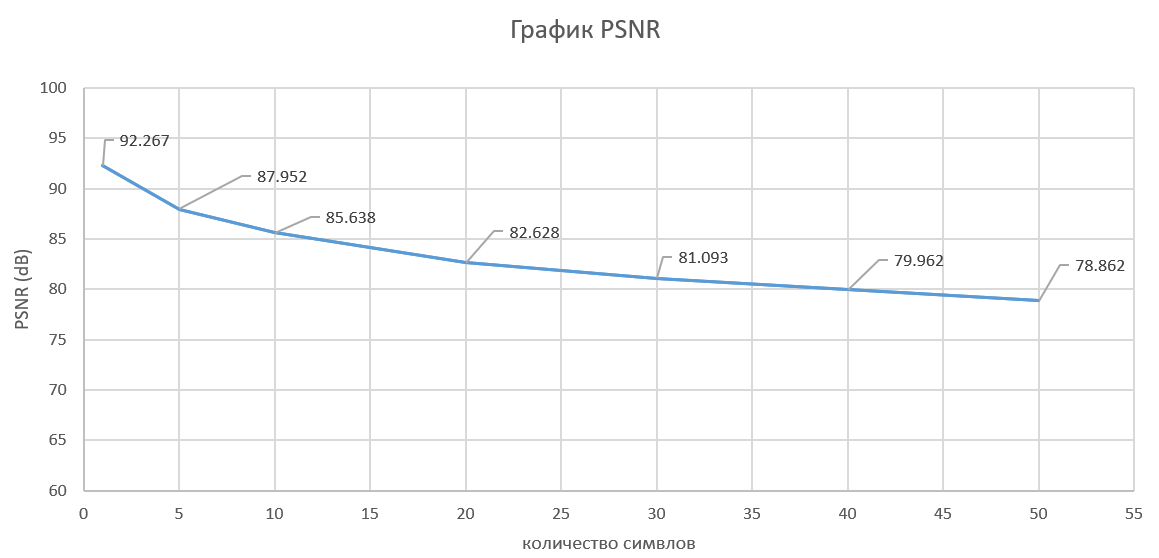
Мы построим графику значений PSNR исходного изображения и изображения с информацией, встроенные по наростающей одно слово, 5, 10, 20, 30, 40,50 до конца текстового файла.



Рисунка 6. Запуск программы “psnr.py” для 7 разных случаи количества символов в картинке

|  |  |
| --- | --- |
| Количество симвлов | PSNR (in dB) |
| 1 | 92.267 |
| 5 | 87.952 |
| 10 | 85.638 |
| 20 | 82.628 |
| 30 | 81.093 |
| 40 | 79.962 |
| 50 | 78.862 |

Таблица 1. Количества симвлов и значения PSNR



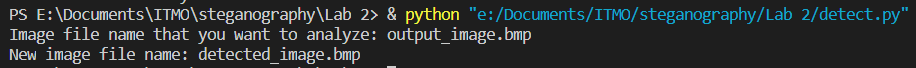
Рисунка 7. График значения PSNR и количества символов

**2. Зависимость значения PSNR от размера встроенного текста**:

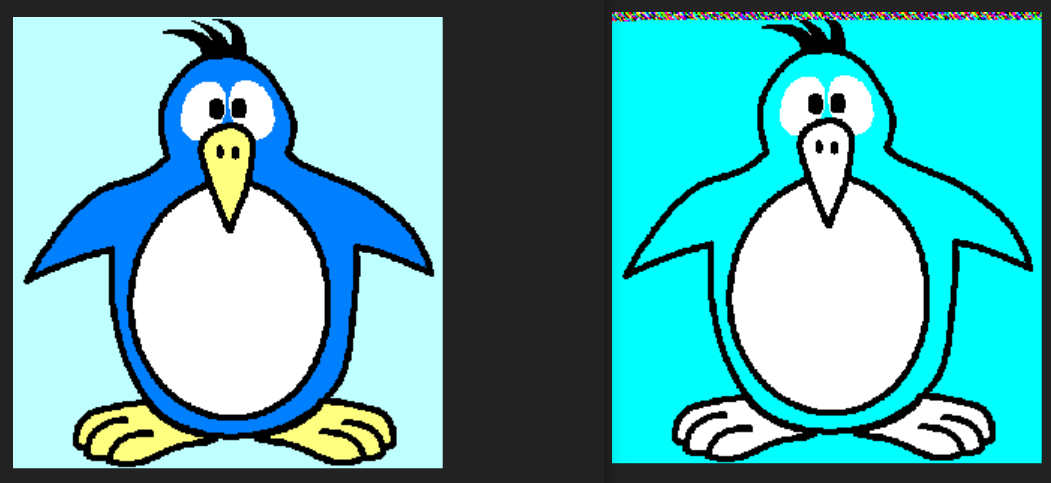
Можно видеть, что значение PSNR снижается, когда мы вставляем больше символа.

**3. Проведение простого атака**:

Для проведения атака, используется программа “detect.py”, если значение НЗБ канала равно 1 программа устанавливает значение канала в 255, если значение НЗБ канала равно 0 устанавливает значение канала в 1. После этого программа сохраняет новое изображение.



Рисунка 8. Запуск программы “detect.py”



Рисунка 9. Разница между “output\_image.bmp” и “detected\_image.bmp”

На основании разницы между 2 изображения. Можно делать вывод, что “output\_image.bmp” содержает сообщение. Потому что буквы a, b, c, d,… , o имеют повторение “0110”, буквы p, q, r, s … , z имеют повторение “0111”. Такие повторения призвели к повторению значения каналов RGB в изображении.

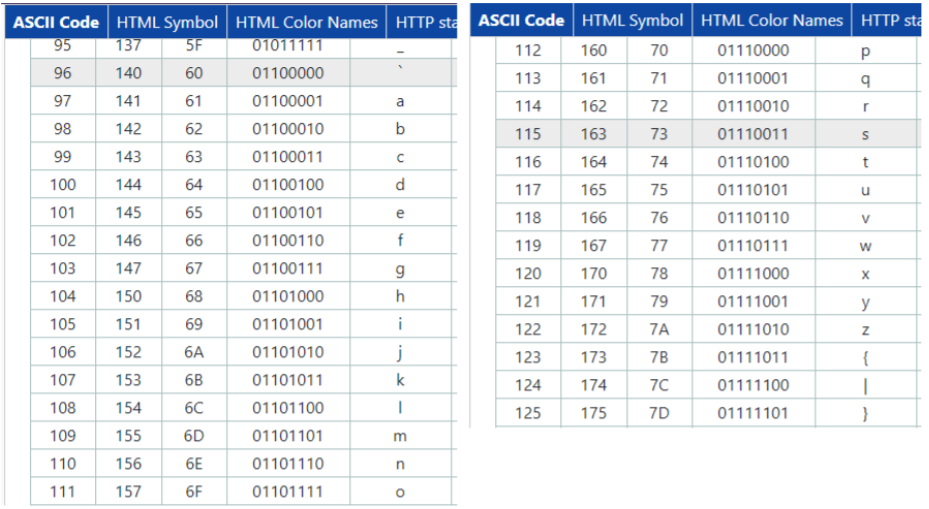
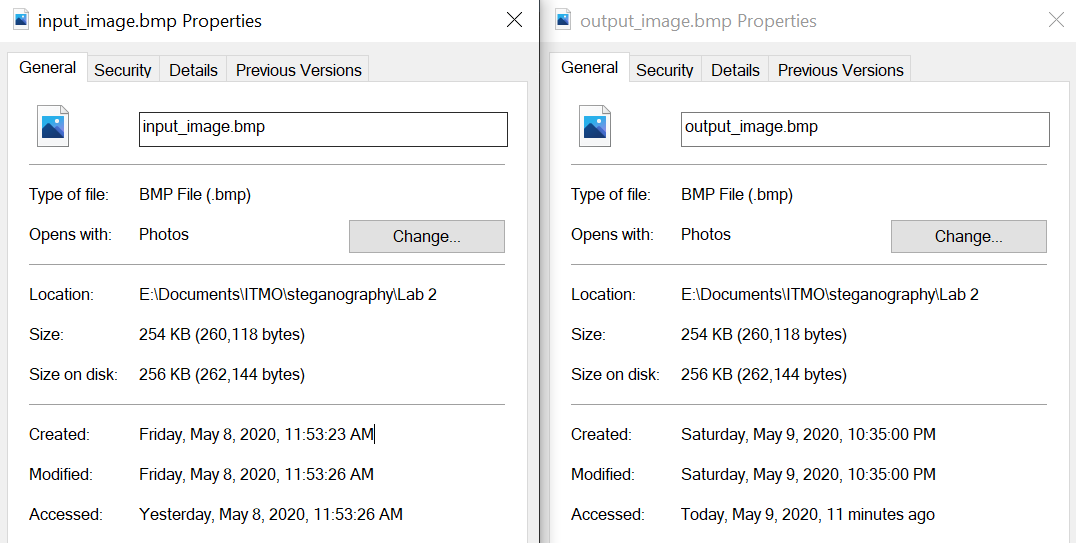


Таблица 2. Значения букв на ASCII таблице

**4. Оценка целесообразности**:



Рисунка 10. Размер изображение до и после встраивания не изменяется

Можно сделать вывод, что метод замены наименьшего значащего бита целесообразен по следующим причинам:

• Можно встраивать большое сообщение (до 1/8 размера исходного изображения).

• Размер стегоконтейнера остается таким же, как размер контейнера.

• Стегоконтейнер неразличимым невооруженным глазом.

# Вывод

При выполнении данной лабораторной работы мною был изучен метод НЗБ стеганографии. Я научился применять его и проводить последующую оценку их применению. По результатам работы были сделаны следующие выводы:

Обнаружение метода НЗБ изображения невооруженным глазом или сравнением размера является невозможным, даже с специальным средствам при отсутствии исходного контейнера довольно сложно. Если я захочу в будущем поместить стего в изображении, то, скорее всего, воспользуюсь методом НЗБ, потому что у него высокая пропускная способность и довольная надежность.

**Список использованной литературы**

1. <https://en.wikipedia.org/wiki/Bit_numbering> - Bit numbering

2. <https://www.ijltet.org/wp-content/uploads/2015/02/60.pdf> - Steganography in Images Using LSB Technique

3. <https://en.wikipedia.org/wiki/BMP_file_format> - BMP file format

4. <https://ru.wikipedia.org/wiki/RGB> - RGB

5. <https://en.wikipedia.org/wiki/Peak_signal-to-noise_ratio> - PSNR (Peak signal-to-noise ratio)

# Приложение

**Программа image\_stegano.py**:

# Python program implementing image-based steganography

from PIL import Image

def str\_2\_bin(secret\_message):

binary\_message = ''

for character in secret\_message:

binary\_message += bin(ord(character))[2:].zfill(8)

return binary\_message

def bin\_2\_str(binary\_message):

secret\_message = ''

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

secret\_message += chr(int(binary\_message[i:i+8], 2))

return secret\_message

def encode(filename\_in, secret\_message, filename\_out):

binary\_message = str\_2\_bin(secret\_message)

binary\_length = bin(len(secret\_message))[2:].zfill(8)

total\_data = binary\_length + binary\_message

#print(total\_data)

image = Image.open(fp=filename\_in, mode='r')

width, heigth = image.size

data\_index = 0

total\_data\_length = len(total\_data)

#while(data\_index < total\_data\_length):

for y in range(heigth): # y

for x in range(width): # x

pixel = list(image.getpixel((x, y))) # x, y

for k in range(len(pixel)):

if data\_index < total\_data\_length:

# Replace the LSB value

new\_pixel = int(bin(pixel[k])[2:].zfill(8)[:-1] + total\_data[data\_index], 2)

pixel[k] = new\_pixel

data\_index += 1

else:

break

image.putpixel((x, y), tuple(pixel))

image.save(filename\_out)

image.close()

def decode(filename\_in):

image\_in = Image.open(filename\_in, mode='r')

width, height = image\_in.size

# Decode the first 8 bits to get the length (a number of characters) of the secret message

binary\_length = ''

binary\_length\_index = 0

for y in range(height):

for x in range(width):

pixel = image\_in.getpixel((x, y))

for i in range(len(pixel)):

if binary\_length\_index < 8:

if pixel[i] % 2 == 0:

binary\_length += '0'

binary\_length\_index += 1

elif pixel[i] % 2 == 1:

binary\_length += '1'

binary\_length\_index += 1

else:

break

length = int(binary\_length, 2)

# Continue decode {length} characters of the secret message, skip the first 8 bits

binary\_message = ''

binary\_message\_index = 0

for y in range(height):

for x in range(width):

pixel = image\_in.getpixel((x, y))

for i in range(len(pixel)):

if binary\_message\_index < 8:

binary\_message\_index += 1

continue

elif binary\_message\_index >= 8 and len(binary\_message) < length \* 8:

if pixel[i] % 2 == 0:

binary\_message += '0'

elif pixel[i] % 2 == 1:

binary\_message += '1'

else:

break

secret\_message = bin\_2\_str(binary\_message)

return secret\_message

# The main program

if \_\_name\_\_ == '\_\_main\_\_':

operation = input('1. Encode, 2. Decode: ')

if operation == '1':

print('ENCODING ...')

input\_text = input('Input text file (input\_text.txt): ')

src\_image = input('Source image file (input\_image.bmp): ')

dst\_image = input('Destination image file (output\_image.bmp): ')

f = open(file=input\_text, mode='r')

secret\_message = f.read()

f.close()

encode(src\_image, secret\_message, dst\_image)

print("Encoded successfully")

elif operation == '2':

print('DECODING ...')

encoded\_image = input('Input image file (output\_image.bmp): ')

output\_text = input('Output text file (output\_text.txt): ')

secret\_message = decode(encoded\_image)

f = open(file=output\_text, mode='w')

f.write(secret\_message)

f.close()

print("Decoded successfully")

else:

exit()

**Программа psnr.py**:

# Calculate PSNR of the original image and the stegcontainer image

from PIL import Image

import math

def psnr(input\_image, output\_image):

output\_image = Image.open(output\_image, mode='r')

width, height = output\_image.size

# Decode the first 8 bits to get the length (a number of characters) of the secret message

binary\_length = ''

binary\_length\_index = 0

for y in range(height):

for x in range(width):

pixel = output\_image.getpixel((x, y))

for i in range(len(pixel)):

if binary\_length\_index < 8:

if pixel[i] % 2 == 0:

binary\_length += '0'

binary\_length\_index += 1

elif pixel[i] % 2 == 1:

binary\_length += '1'

binary\_length\_index += 1

else:

break

# Number of channels need to be calculated

channel\_number = int(binary\_length, 2) \* 8 + 8

# Calculate PSNR of the {length \* 8} bits, skip the first 8 bits

input\_image = Image.open(input\_image, mode='r')

current\_channel\_number = 0

I\_K\_2 = 0

for y in range(height):

for x in range(width):

output\_pixel = output\_image.getpixel((x, y))

input\_pixel = input\_image.getpixel((x, y))

for i in range(len(pixel)):

if current\_channel\_number < channel\_number:

subtraction = input\_pixel[i] - output\_pixel[i]

if (subtraction != 0):

I\_K\_2 += 1

current\_channel\_number += 1

else:

break

mse = I\_K\_2 / (width \* height \* 3)

psnr = 10 \* math.log(pow(255, 2) / mse, 10)

return psnr

# The main program

if \_\_name\_\_ == '\_\_main\_\_':

# Change the parameters

print(psnr('input\_image.bmp', 'output\_image50.bmp'))

**Программа detect.py**:

# Detect image-based steganography

from PIL import Image

def detect(encoded\_image, detected\_image):

image = Image.open(fp=encoded\_image, mode='r')

width, heigth = image.size

for y in range(heigth):

for x in range(width):

pixel = list(image.getpixel((x, y)))

for k in range(len(pixel)):

if pixel[k] % 2 == 0:

pixel[k] = 1

elif pixel[k] % 2 == 1:

pixel[k] = 255

image.putpixel((x, y), tuple(pixel))

image.save(detected\_image)

image.close()

# The main program

if \_\_name\_\_ == '\_\_main\_\_':

sample\_image = input('Image file name that you want to analyze: ')

detected\_image = input('New image file name: ')

detect(sample\_image, detected\_image)