Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Лабораторная работа № 14**

«ИССЛЕДОВАНИЕ СТЕГАНОГРАФИЧЕСКОГО МЕТОДА НА ОСНОВЕ ПРЕОБРАЗОВАНИЯ НАИМЕНЕЕ ЗНАЧАЩИХ БИТОВ»

Выполнил:

Студент: Дащинский М.Л..

ФИТ 3 курс 4 группа

Преподаватель: Сазонова Д.В.

Минск 2023

1. **Описание приложения**

Приложение написано на языке программирования Python, в котором реализован метод НЗБ.

1. **Методика выполнения расчетов**

В данной лабораторной работе была поставлена цель создания приложения, которое реализовывает метод НЗБ. На листинге 2.1 представлены функции, реализующие данную функциональность.

from PIL import Image

from supp\_function import open\_image, extract\_information

class Error(Exception):

*"""Базовый класс для других исключений"""*

    pass

class SmallImageSize(Error):

*"""Вызывается, когда размеры фотографии маленькие"""*

    pass

def encrypt(image\_path: str, embedded\_message: str, embedding\_mode: int):

    try:

        rgb\_image\_np = open\_image(image\_path)

        size\_image = rgb\_image\_np.shape[0:2]

        if len(embedded\_message) > size\_image[0] \* size\_image[1]:

            raise SmallImageSize

        rgb\_image\_np = rgb\_image\_np.reshape(

            (rgb\_image\_np.shape[0] \* rgb\_image\_np.shape[1], 3))

        if embedding\_mode == 1:

            for i in range(len(embedded\_message)):

                rgb\_image\_np[i][2] = 1 & int(embedded\_message[i])

        elif embedding\_mode == 2:

            random\_positions = extract\_information(

                size\_image, len(embedded\_message))

            for i in range(len(embedded\_message)):

                rgb\_image\_np[random\_positions[i]][2] = 1 & int(

                    embedded\_message[i])

        rgb\_image\_np = rgb\_image\_np.reshape((size\_image[0], size\_image[1], 3))

        new\_image = Image.fromarray(rgb\_image\_np, 'RGB')

        encrypted\_image\_path = R"D:\University\_BSTU\Labs\КМЗИ\Лабы\Lab-14\encrypted.png"

        new\_image.save(encrypted\_image\_path)

        return len(embedded\_message)

    except SmallImageSize:

        print(f'Размеры изображения малы для встраивания')

        return 0

def decrypt(image\_path: str, len\_embedded\_message: int, embedding\_mode: int):

    rgb\_image\_np = open\_image(image\_path)

    size\_image = rgb\_image\_np.shape[0:2]

    rgb\_image\_np = rgb\_image\_np.reshape(

        (rgb\_image\_np.shape[0] \* rgb\_image\_np.shape[1], 3))

    decrypt\_binary\_message = ''

    if embedding\_mode == 1:

        for i in range(len\_embedded\_message):

            decrypt\_binary\_message += str(rgb\_image\_np[i][2] & 1)

    elif embedding\_mode == 2:

        random\_positions = extract\_information(

            size\_image, len\_embedded\_message)

        for i in random\_positions:

            decrypt\_binary\_message += str(rgb\_image\_np[i][2] & 1)

    decrypt\_message = ''.join([chr(int(decrypt\_binary\_message[i \* 16: (i + 1) \* 16], 2))

                               for i in range(len(decrypt\_binary\_message) // 16)])

    return decrypt\_message

def get\_LSB(file\_for\_shades: str):

    rgb\_image\_np = open\_image(file\_for\_shades)

    size\_image = rgb\_image\_np.shape[0], rgb\_image\_np.shape[1]

    rgb\_image\_np\_red = rgb\_image\_np.copy().reshape(

        (rgb\_image\_np.shape[0] \* rgb\_image\_np.shape[1], 3))

    rgb\_image\_np\_green = rgb\_image\_np.copy().reshape(

        (rgb\_image\_np.shape[0] \* rgb\_image\_np.shape[1], 3))

    rgb\_image\_np\_blue = rgb\_image\_np.copy().reshape(

        (rgb\_image\_np.shape[0] \* rgb\_image\_np.shape[1], 3))

    rgb\_image\_np\_rgb = rgb\_image\_np.copy().reshape(

        (rgb\_image\_np.shape[0] \* rgb\_image\_np.shape[1], 3))

    for i in range(rgb\_image\_np.shape[0] \* rgb\_image\_np.shape[1]):

        current\_red\_LSB = rgb\_image\_np\_red[i][0] & 1

        rgb\_image\_np\_red[i] = list(

            map(lambda x: x \* current\_red\_LSB, [255, 255, 255]))

        current\_green\_LSB = rgb\_image\_np\_green[i][1] & 1

        rgb\_image\_np\_green[i] = list(

            map(lambda x: x \* current\_green\_LSB, [255, 255, 255]))

        current\_blue\_LSB = rgb\_image\_np\_blue[i][2] & 1

        rgb\_image\_np\_blue[i] = list(

            map(lambda x: x \* current\_blue\_LSB, [255, 255, 255]))

        rgb\_image\_np\_rgb[i] = rgb\_image\_np\_red[i][0], rgb\_image\_np\_green[i][1], rgb\_image\_np\_blue[i][2]

    rgb\_image\_np\_red = rgb\_image\_np\_red.reshape(

        (size\_image[0], size\_image[1], 3))

    rgb\_image\_np\_green = rgb\_image\_np\_green.reshape(

        (size\_image[0], size\_image[1], 3))

    rgb\_image\_np\_blue = rgb\_image\_np\_blue.reshape(

        (size\_image[0], size\_image[1], 3))

    rgb\_image\_np\_rgb = rgb\_image\_np\_rgb.reshape(

        (size\_image[0], size\_image[1], 3))

    new\_image\_red = Image.fromarray(rgb\_image\_np\_red, 'RGB')

    new\_image\_green = Image.fromarray(rgb\_image\_np\_green, 'RGB')

    new\_image\_blue = Image.fromarray(rgb\_image\_np\_blue, 'RGB')

    new\_image\_rgb = Image.fromarray(rgb\_image\_np\_rgb, 'RGB')

    encrypted\_image\_path\_red = R"D:\University\_BSTU\Labs\КМЗИ\Лабы\Lab-14\red\_shades.png"

    encrypted\_image\_path\_green = R"D:\University\_BSTU\Labs\КМЗИ\Лабы\Lab-14\green\_shades.png"

    encrypted\_image\_path\_blue = R"D:\University\_BSTU\Labs\КМЗИ\Лабы\Lab-14\blue\_shades.png"

    encrypted\_image\_path\_rgb = R"D:\University\_BSTU\Labs\КМЗИ\Лабы\Lab-14\rgb\_shades.png"

    new\_image\_red.save(encrypted\_image\_path\_red)

    new\_image\_green.save(encrypted\_image\_path\_green)

    new\_image\_blue.save(encrypted\_image\_path\_blue)

    new\_image\_rgb.save(encrypted\_image\_path\_rgb)

def task\_1():

    message = open('text.txt', 'r', encoding='utf8').read()[:100]

    embedded\_message = ''.join(

        [bin(ord(i))[2:].rjust(16, '0') for i in message])

    file\_for\_embedding = R"embedded\_image.jpg"

    file\_to\_readout = R"D:\University\_BSTU\Labs\КМЗИ\Лабы\Lab-14\encrypted.png"

    len\_embedded\_message = encrypt(file\_for\_embedding, embedded\_message, 1)

    decrypt\_message = decrypt(file\_to\_readout, len\_embedded\_message, 1)

    print(f'----- Встраивомое сообщение -----\n{message}\n'

          f'----- Считанное сообщение -----\n{decrypt\_message}\n'

          f'----- Сообщение считано {"верно!" if message == decrypt\_message else "не верно!"} -----')

def task\_2():

    file\_for\_shades = R"embedded\_image.jpg"

    get\_LSB(file\_for\_shades)

def task\_3():

    message = open('text.txt', 'r', encoding='utf8').read()[:100]

    embedded\_message = ''.join(

        [bin(ord(i))[2:].rjust(16, '0') for i in message])

    file\_for\_embedding = R"embedded\_image.jpg"

    file\_to\_readout = R"D:\University\_BSTU\Labs\КМЗИ\Лабы\Lab-14\encrypted.png"

    len\_embedded\_message = encrypt(file\_for\_embedding, embedded\_message, 2)

    decrypt\_message = decrypt(file\_to\_readout, len\_embedded\_message, 2)

    print(f'----- Встраиваемое сообщение -----\n{message}\n'

          f'----- Считанное сообщение -----\n{decrypt\_message}\n'

          f'----- Сообщение считано {"верно!" if message == decrypt\_message else "не верно!"} -----')

from PIL import Image

import numpy as np

def open\_image(filename: str):

    if filename:

        image = Image.open(filename)

        img\_rgb = image.convert('RGB')

        rgb\_image\_np = np.copy(img\_rgb)

        return rgb\_image\_np

def extract\_information(list\_image, count\_number: int):

    a = list\_image[0] + 19

    b = list\_image[1] + 19

    current\_x = 3

    n = list\_image[0] \* list\_image[1]

    random\_numbers = []

    for i in range(count\_number):

        current\_x = (a \* current\_x + b) % n

        random\_numbers.append(current\_x)

    return random\_numbers

Листинг 2.1 –код программы, реализующие заданную ранее функциональность

**3. Результаты работы приложения**

Для выполнения расчетов достаточно необходимо запустить приложение. Рисунок 3.1 показывают требуемые в данной лабораторной работе результаты. Также будут созданые необходимые файлы во время работы программы.

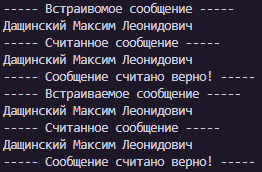


Рисунок 3.1 – Результат работы

**Вывод**

В ходе лабораторной работы было разработано приложение для реализации метода НЗБ.