Національний технічний університет України
«Київський політехнічний інститут»
Інститут Прикладного системного аналізу
Кафедра Системного проектування

Контрольна робота

з дисципліни «**Комп'ютерні мережі**»

«Дослідження стеганографічних методів приховування даних»

Виконав:
студент групи ДА-82
факультету «ІПСА»
Муравльов А.Д.
Варіант 18

Київ – 2021

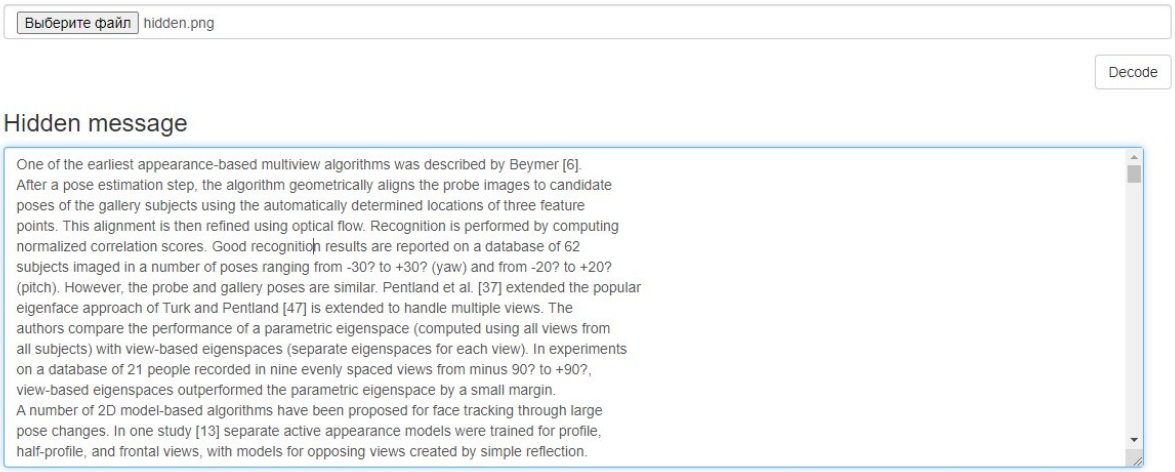
Хід роботи

1. Вибрати підходящий контейнер у вигляді файлу зображення в форматі bmp для приховування текстового файлу і за допомогою програми S-Tools 4.0 виконати приховування файлу, використовуючи один з доступних алгоритмів шифрування. Вилучити файли з контейнера і порівняти їх з оригіналом. Зберегти отримані результати.

Скористатись програмою S-Tools не вдалося – програма видавала повідомлення Out of memory при спробі приховати текст в картинці, тому скористаємось сайтом, який надає схожий функціонал.

Оригінал (порожній контейнер)

Контейнер з прихованим текстом

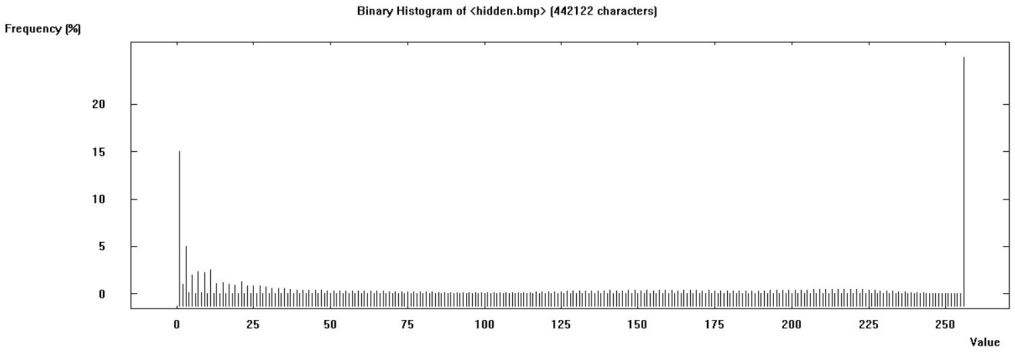


Результат декодування

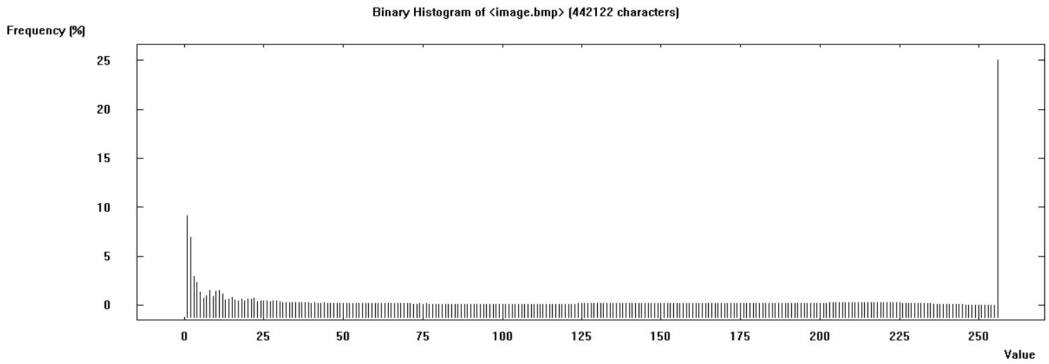
2. За допомогою засобів пакету CrypTool виконати статистичний аналіз порожнього контейнера і контейнера з прихованими даними. Для цього знайти ентропії контейнерів і гістограми контейнерів. Порівняти отримані дані, результати зберегти.

Ентропія

Порожній контейнер	Контейнер з прихованим текстом
5.23	3.86



Гістограма контейнера з прихованим текстом



Гістограма порожнього контейнера

3.*Для обраного при виконанні п. 2 контейнера розрахувати допустимий обсяг приховуваних даних за методом LSB і порівняти його з можливостями програми S-Tools 4.0.*

Обрали контейнер розміром 325 х 340 пікселів. Зображення кольорове, тому кількість байт в зображенні 325 х 340 х 3 = 331500.

В залежності від того, яку кількість молодших бітів будемо змінювати, буде змінюватись розмір інформації, яку хочемо зашифрувати.

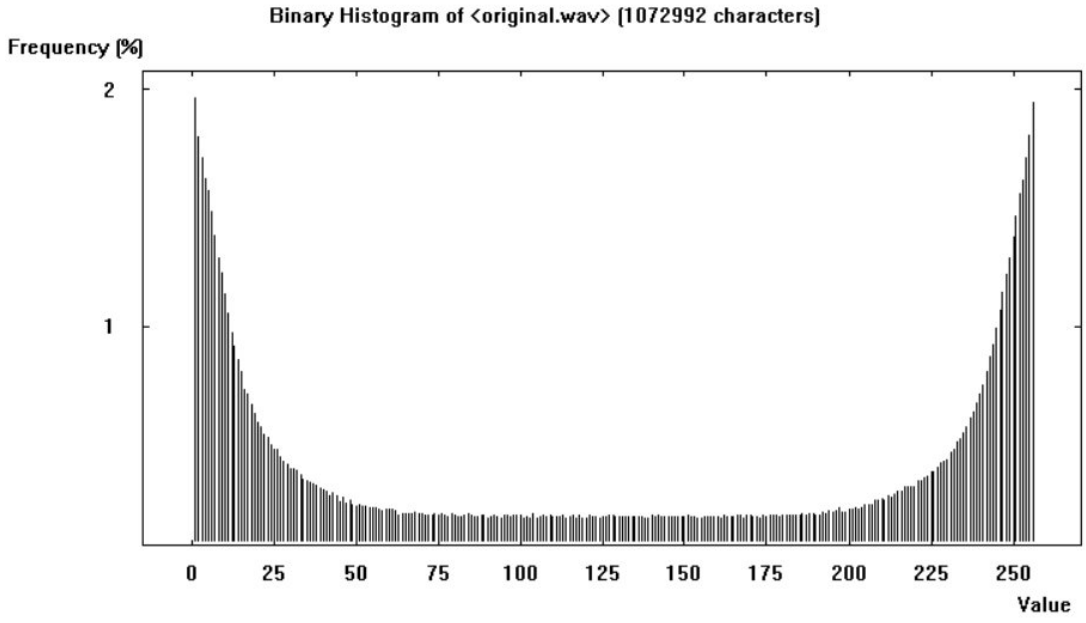
Так, якщо вирішуємо змінювати лише один молодший біт, то максимальний розмір приховуваних даних буде становити 975360 / 8 = 41438 байт.

Аналогічним чином розраховується обсяг для випадків, коли кількість змінюваних молодших бітів зростає.

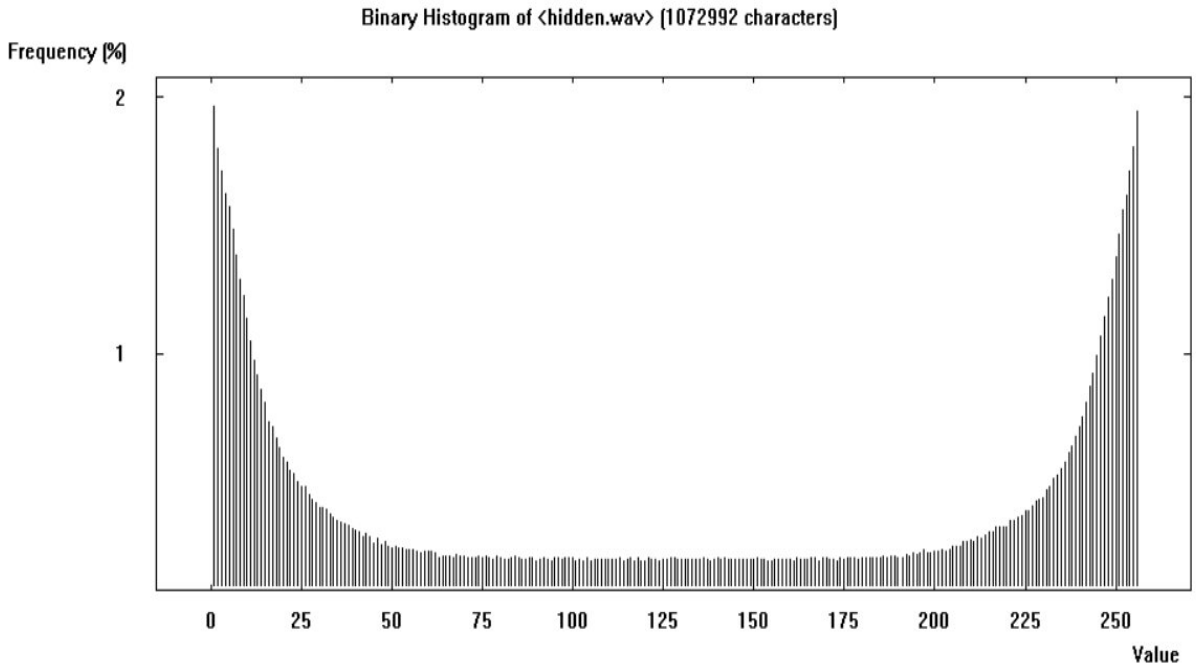
4.Обрати підходящий контейнер у вигляді звукового файлу. Виконати приховування файлу, використовуючи один з доступних алгоритмів шифрування. Вилучити файли з контейнера і порівняти їх з оригіналом. Виконати статистичний аналіз порожнього контейнера і контейнера з прихованими даними. Для цього знайти ентропії контейнерів і гістограми контейнерів.

Ентропія

Порожній контейнер	Контейнер з прихованим текстом
7.33	7.35



Гістограма порожнього контейнера



Гістограми контейнера з прихованим текстом

5.Написати програму для приховування (і вилучення) даних в файлах напівтонових (у відтінках сірого) зображень за методом LSB (без шифрування) з вісьма біт на один піксель, налагодити її і перевірити працездатність на заданому текстовому файлі.

Вихідний код програми:

```
from PIL import Image

def genData(data):
        newd = []
        for i in data:
            newd.append(format(ord(i), '08b'))
        return newd

def modPix(pix, data):

    datalist = genData(data)
    lendata = len(datalist)
    imdata = iter(pix)

    for i in range(lendata):
        pix = [value for value in imdata.__next__()[:3] +
                                imdata.__next__()[:3] +
                                imdata.__next__()[:3]]
        for j in range(0, 8):
            if (datalist[i][j] == '0' and pix[j]% 2 != 0):
                pix[j] -= 1
```

```
            elif (datalist[i][j] == '1' and pix[j] % 2 == 0):
                if(pix[j] != 0):
                    pix[j] -= 1
                else:
                    pix[j] += 1
        if (i == lendata - 1):
            if (pix[-1] % 2 == 0):
                if(pix[-1] != 0):
                    pix[-1] -= 1
                else:
                    pix[-1] += 1
        else:
            if (pix[-1] % 2 != 0):
                pix[-1] -= 1
        pix = tuple(pix)
        yield pix[0:3]
        yield pix[3:6]
        yield pix[6:9]
def encode_enc(newimg, data):
    w = newimg.size[0]
    (x, y) = (0, 0)
    for pixel in modPix(newimg.getdata(), data):
        newimg.putpixel((x, y), pixel)
        if (x == w - 1):
            x = 0
            y += 1
        else:
            x += 1
def encode():
    img = input("Enter path to container image: ")
    image = Image.open(img, 'r')

    text_path = input("Enter path to data to be encoded: ")

    data = ""
    with open(text_path, 'r') as inputFile:
        data = ' '.join(inputFile.readlines())

    newimg = image.copy()
    encode_enc(newimg, data)

    new_img_name = input("Enter path to new image: ")
    newimg.save(new_img_name, str(new_img_name.split(".")[1].upper()))

def decode():
    img = input("Enter path to container image: ")
    image = Image.open(img, 'r')
    data = ''
    imgdata = iter(image.getdata())

    while (True):
```

```
        pixels = [value for value in imgdata.__next__()[:3] +
                                imgdata.__next__()[:3] +
                                imgdata.__next__()[:3]]
        binstr = ''
        for i in pixels[:8]:
            if (i % 2 == 0):
                binstr += '0'
            else:
                binstr += '1'
        data += chr(int(binstr, 2))
        if (pixels[-1] % 2 != 0):
            return data

def main():
    a = int(input("Choose action type\n"
                        "1. Encode\n2. Decode\n"))
    if (a == 1):
        encode()
    elif (a == 2):
        print("Decoded Text: " + decode())
    else:
        raise Exception("Enter correct input")

if __name__ == '__main__' :
    main()
```

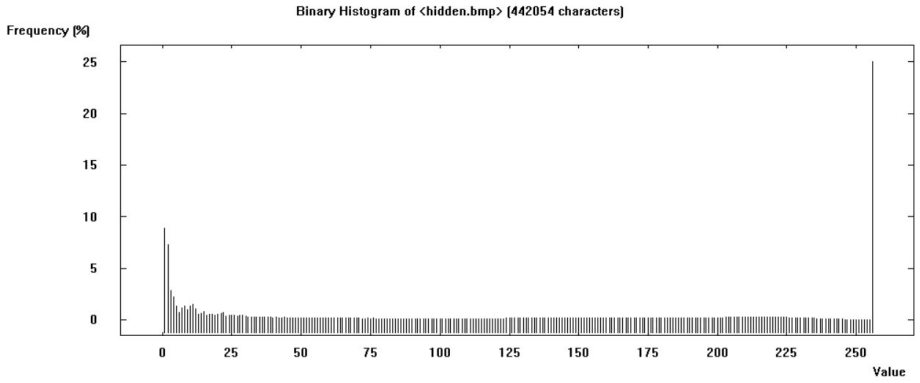
```
PS D:\Labs\Andrey\bis5\code> python main.py
Choose action type
1. Encode
2. Decode
1
Enter path to container image: D:\Labs\Andrey\bis5\images\image.bmp
Enter path to data to be encoded: D:\Labs\Andrey\bis5\texts\v18.txt
Enter path to new image: D:\Labs\Andrey\bis5\images\hidden.bmp
PS D:\Labs\Andrey\bis5\code> python main.py
Choose action type
1. Encode
2. Decode
2
Enter path to container image: D:\Labs\Andrey\bis5\images\hidden.bmp
Decoded Text: One of the earliest appearance-based multiview algorithms was described by Beymer [6].
 After a pose estimation step, the algorithm geometrically aligns the probe images to candidate
 poses of the gallery subjects using the automatically determined locations of three feature
 points. This alignment is then refined using optical flow. Recognition is performed by computing
 normalized correlation scores. Good recognition results are reported on a database of 62
 subjects imaged in a number of poses ranging from -30? to +30? (yaw) and from -20? to +20?
```

Робота програми в консолі

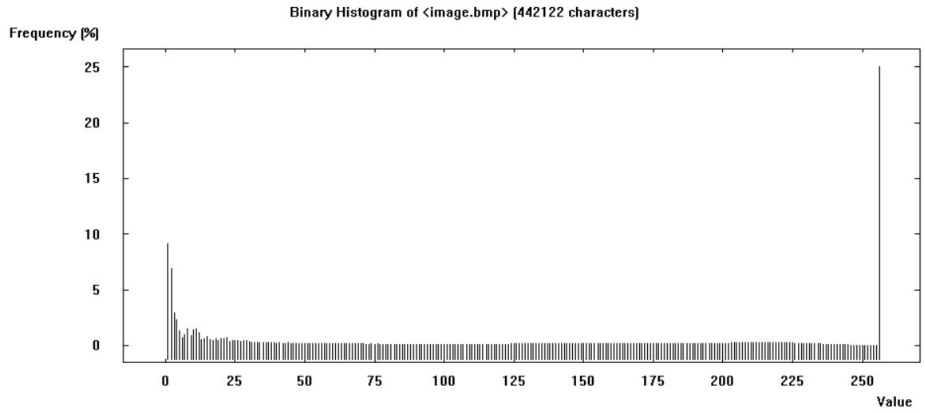
7. За допомогою засобів пакету CrypTool виконати статистичний аналіз порожнього контейнера і контейнера з прихованими даними, отриманого за допомогою розробленої програми. Для цього знайти ентропії контейнерів і гістограми контейнерів. Порівняти отримані дані, результати зберегти. Зробити висновки про можливість використання гістограмного аналізу для виявлення прихованих даних.

Ентропія

Порожній контейнер	Контейнер з прихованим текстом
5.87	5.87



Гістограма контейнеру з прихованим текстом



Гістограма порожнього контейнеру

Порівнявши гістограми можемо зробити висновок, шо два сусідніх значення зустрічаються з однаковою частотою. Таким чином можна зазначити, що при використанні методу LSB можливо виявити факт вбудування певної інформації в файл.

Висновки

В ході роботи було досліджено методи стеганографії, які можуть застосовуватись для вбудовування інформації до графічних та звукових файлів, при цьому цей факт буде непомітний для людських органів сприйняття. Також було реалізовано програму на мові програмування Python, яка реалізовує стеганографічний метод LSB, перевірено її працездатність та проведено аналіз стійкості даного алгоритму до стеганоаналізу.