НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ  
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

Інститут прикладного системного аналізу  
Кафедра системного проектування

Лабораторна робота №5  
з дисципліни “**Безпека інформаційних систем**”  
на тему: “Дослідження стеганографічних методів приховування даних”

Виконав:  
студент групи ДА-82  
факультету “ІПСА”  
Муравльов А. Д.  
Варіант 18

Київ — 2021

[**Мета роботи** 2](#_Toc1)

[**Завдання роботи** 2](#_Toc2)

[**Хід роботи** 3](#_Toc3)

[**1. Приховування файлу за допомогою програмного забезпечення** 3](#_Toc4)

[**2. Статистичний аналіз у CrypTool** 4](#_Toc5)

[**3. Підрахунок допустимого обсягу приховуваних даних** 5](#_Toc6)

[**4. Порівняння порожнього контейнера з контейнером з прихованим текстом** 5](#_Toc7)

[**5. Програмна реалізація** 6](#_Toc8)

[**6. Аналіз контейнерів за допомогою пакета Cryptool** 10](#_Toc9)

[**ВИСНОВКИ** 12](#_Toc10)

# **Мета роботи**

Дослідити стеганографічні методи приховування даних та їх реалізація.

# **Завдання роботи**

1. Обрати варіант текстового файлу (див. Додаток до лабораторної роботи № 1) згідно з порядковим номером студента в списку академічної групи.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
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.
8. Оформити звіт по роботі.

# **Хід роботи**

# **1. Приховування файлу за допомогою програмного забезпечення**

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.

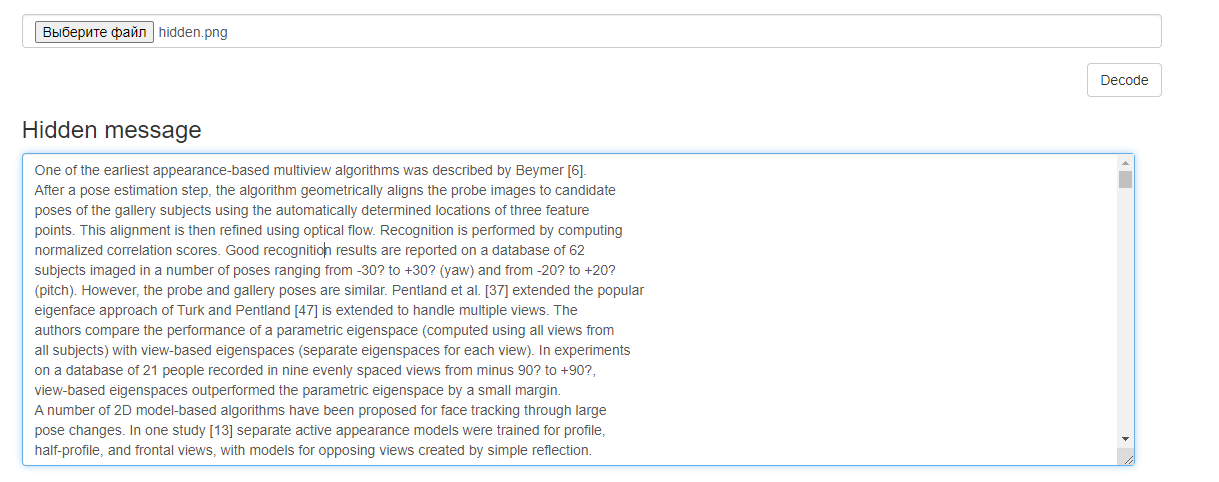
Скористатись програмою S-Tools не вдалося – програма видавала повідомлення Out of memory при спробі приховати текст в картинці, тому скористаємось [сайтом](https://stylesuxx.github.io/steganography), який надає схожий функціонал.



Оригінал (порожній контейнер)



Контейнер з прихованим текстом



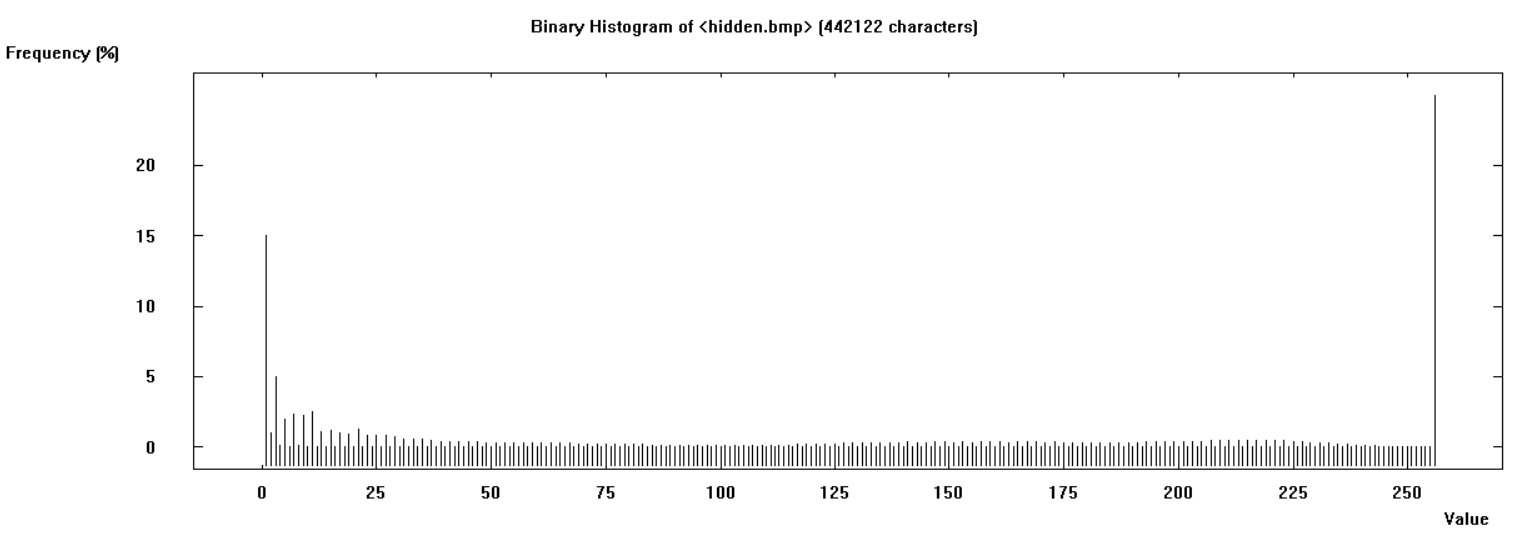
Результат декодування

# **2. Статистичний аналіз у CrypTool**

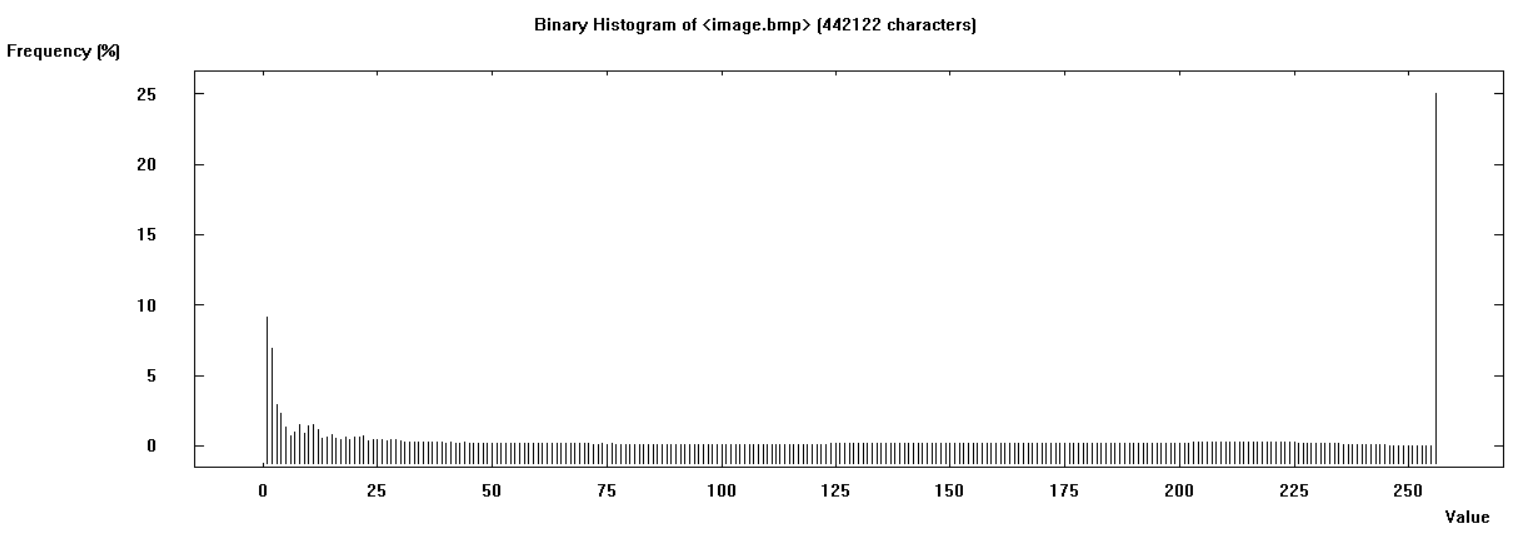
За допомогою засобів пакету CrypTool виконати статистичний аналіз порожнього контейнера і контейнера з прихованими даними. Для цього знайти ентропії контейнерів і гістограми контейнерів. Порівняти отримані дані, результати зберегти.

Ентропія

|  |  |
| --- | --- |
| Порожній контейнер | Контейнер з прихованим текстом |
| 5.23 | 3.86 |



Гістограма контейнера з прихованим текстом



Гістограма порожнього контейнера

# **3. Підрахунок допустимого обсягу приховуваних даних**

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 x 340 x 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 331500/ 8 = 41438 байт.

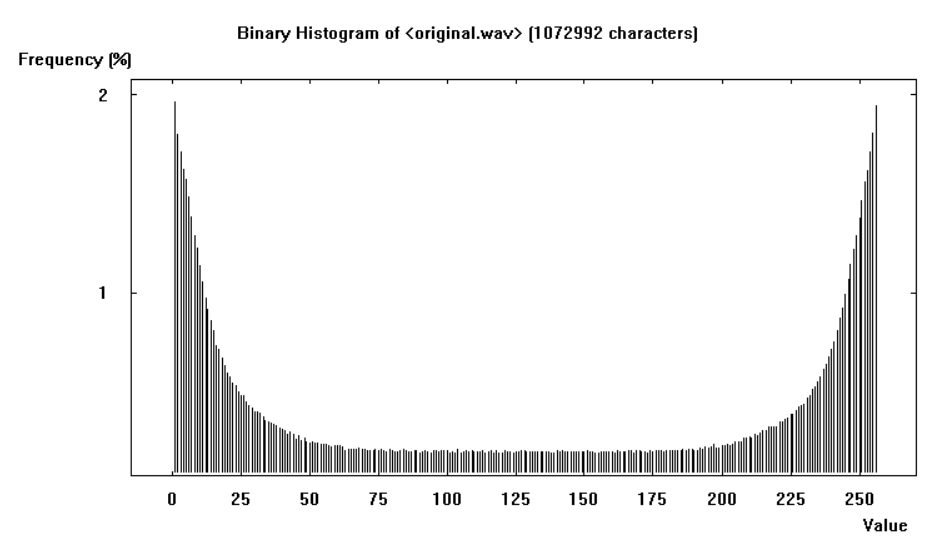
Аналогічним чином розраховується обсяг для випадків, коли кількість змінюваних молодших бітів зростає.

# **4. Порівняння порожнього контейнера з контейнером з прихованим текстом**

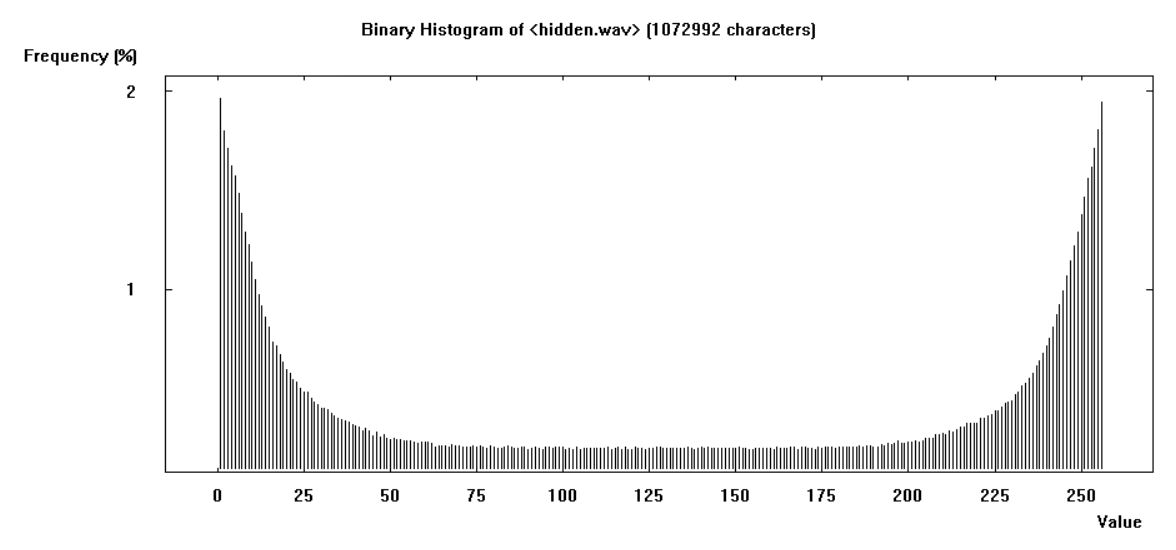
*Обрати підходящий контейнер у вигляді звукового файлу. Виконати приховування файлу, використовуючи один з доступних алгоритмів шифрування. Вилучити файли з контейнера і порівняти їх з оригіналом. Виконати статистичний аналіз порожнього контейнера і контейнера з прихованими даними. Для цього знайти ентропії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

Ентропія

|  |  |
| --- | --- |
| Порожній контейнер | Контейнер з прихованим текстом |
| 7.33 | 7.35 |



Гістограма порожнього контейнера



Гістограма контейнера з прихованим текстом

# **5. Програмна реалізація**

*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):

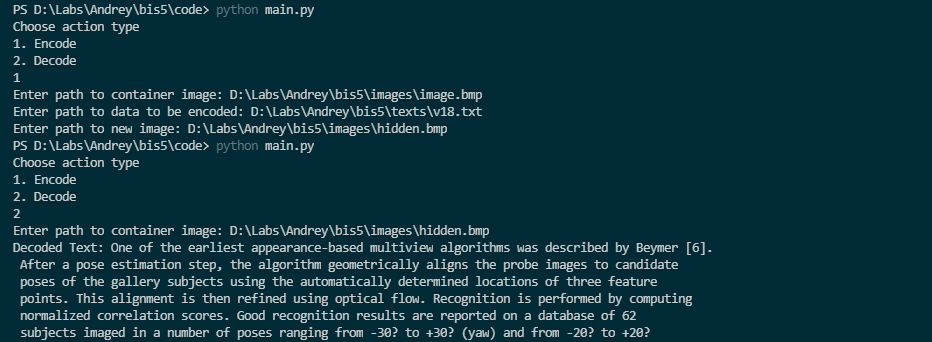
        print("Decoded Text: " + decode())

    else:

        raise Exception("Enter correct input")

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

    main()



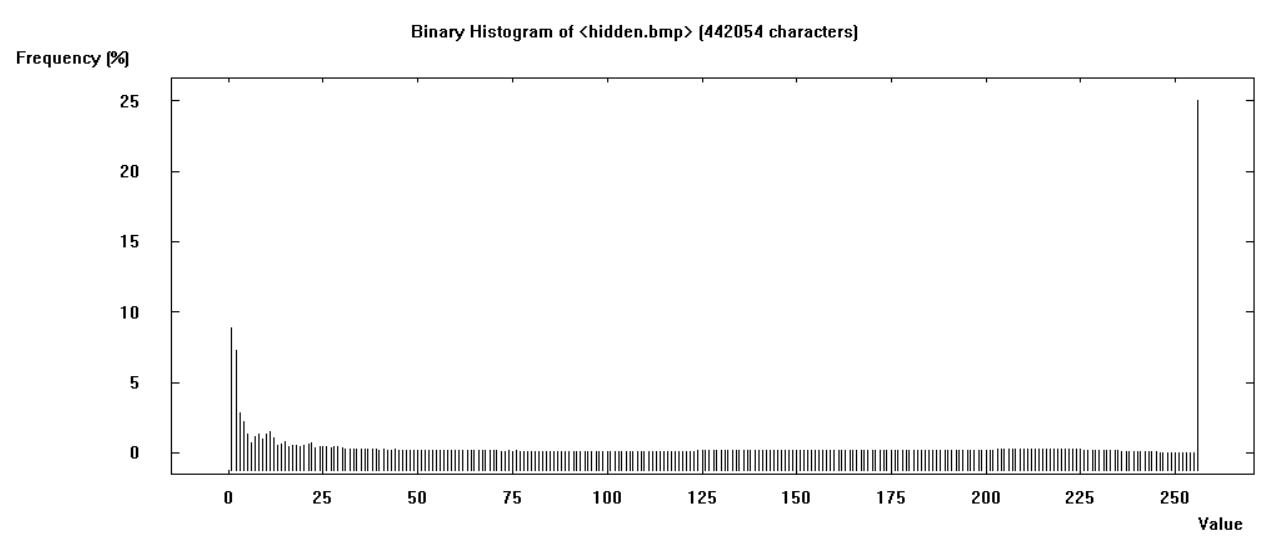
Робота програми в консолі

# **6. Аналіз контейнерів за допомогою пакета Cryptool**

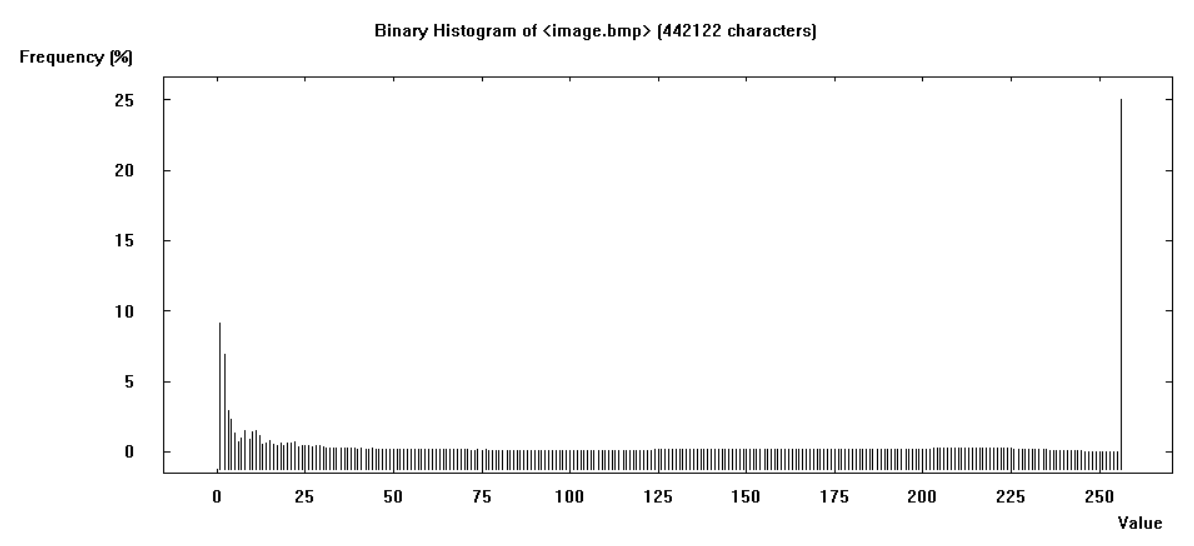
*За допомогою засобів пакету CrypTool виконати статистичний аналіз порожнього контейнера і контейнера з прихованими даними, отриманого за допомогою розробленої програми. Для цього знайти ентропії контейнерів і гістограми контейнерів. Порівняти отримані дані, результати зберегти. Зробити висновки про 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.