

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра інформаційних систем та мереж



Лабораторна робота №4

з дисципліни: «Технології захисту інформації»

на тему: «Шифрування даних методом стеганографії»

Варіант №6

Виконала:

студентка групи ІТ-32

Моляща Ю.А.

Прийняв:

Табачишин Д.Р.

Львів - 2023

Лабораторна робота №4

«Шифрування даних методом стеганографії»

Мета роботи: навчитися опрацьовувати (шифрувати та дешифрувати) файли з прихованими інформаційними повідомленнями.

Завдання на лабораторну роботу.

Завдання 1. Розміщення однобайтних елементів повідомлення в файл графічного формату 24-біти на колір

Хід роботи.

Код програми:

```
import os
from PIL import Image
BITS_PER_BYTE = 8
CHANNELS_PER_PIXEL = 3
def text_to_binary(message):
    binary_data = [ format(ord(char), '08b') for char in message ]
    return binary_data
def binary_to_text(binary):
    message = "".join( chr(code) for code in binary)
    return message
def get_pixel_values(image_path):
    image = Image.open(image_path, 'r')
    pixel_values = list(image.getdata())
    return pixel_values
def encode_pixel(pixel, binary_message, index, byte, counter):
    """Encode a single pixel"""
    if index % BITS_PER_BYTE == 0:
        index = 0

    if len(binary_message) > byte:
        pixel_tuple = ()
        for color_channel in range(CHANNELS_PER_PIXEL):
            if index % BITS_PER_BYTE != 0 or index == 0:
                byte_value = binary_message[byte]
                bit_value = byte_value[index]
                index += 1
                if bit_value == '0':
                    offset = 0 if pixel[color_channel] % 2 == 0 else -1
                elif bit_value == '1':
                    offset = -1 if pixel[color_channel] % 2 == 0 else 0
                pixel_tuple += (pixel[color_channel] + offset,)
            else:
                byte += 1
                if len(binary_message) == byte:
                    offset = 0 if pixel[color_channel] % 2 != 0 else -1
```

```

        else:
            offset = -1 if pixel[color_channel] % 2 != 0 else 0
            pixel_tuple += (pixel[color_channel] + offset,)
    else:
        if counter == 2:
            if pixel[2] % 2 == 0:
                pixel_tuple = pixel[:2] + (pixel[2] - 1,)
            else:
                pixel_tuple = pixel[:2] + (pixel[2],)
            counter = 0
        else:
            pixel_tuple = pixel
            counter += 1

    return pixel_tuple, index, byte, counter

def encode(image_path, message, output_path):
    """Encode the message into the output image"""
    container = Image.open(image_path)
    binary_message = text_to_binary(message)
    if len(binary_message) * BITS_PER_BYTE > container.width * container.height
* CHANNELS_PER_PIXEL:
        raise ValueError("Message is too long for the container.")
    # Number of bits processed (0-7), 8th bit triggers a new byte
    index = 0
    # Number of bytes processed
    byte = 0
    # Counter to check if the last pixel has been reached
    counter = 0

    for y in range(container.height):
        for x in range(container.width):
            pixel = container.getpixel((x, y))
            pixel_tuple, index, byte, counter = encode_pixel(pixel,
binary_message, index, byte, counter)
            container.putpixel((x, y), pixel_tuple)

    container.save(output_path)
    print("Message embedded successfully.")

def decode(output_path):
    """Decode the message hidden in the output image"""
    pixel_values = get_pixel_values(output_path)
    combined_pixel_tuples = triple_pixel_tuple(pixel_values)

    while True:
        data = ''

        for j in range(len(combined_pixel_tuples)):
            byte_char = ''

```

```

        # Extract the first 8 values from the combined tuple
        # and convert to a binary string
        for item in combined_pixel_tuples[j][:8]:
            if item % 2 == 0:
                byte_char += '0'
            else:
                byte_char += '1'
        # Convert the binary string to an integer and then to a character
        # and append to the data string
        data += chr(int(byte_char, 2))

        # Check if the last bit is odd or even
        # If even, return the data string
        if combined_pixel_tuples[j][-1] % 2 != 0:
            return data

def triple_pixel_tuple(pixel_values):
    """Return a list of tuples with 9 values"""
    triple_pixel_tuple = []
    for i in range(0, len(pixel_values), 3):
        tuple1, tuple2, tuple3 = pixel_values[i:i + 3]
        combined_tuple = tuple1 + tuple2 + tuple3
        triple_pixel_tuple.append(combined_tuple)

    return triple_pixel_tuple

def main():
    image_path = 'source/lab4/magma4x6.png'
    output_path = 'source/lab4/output.png'
    message = 'kachow'

    try:
        if not os.path.exists(image_path):
            raise FileNotFoundError(f"The input image '{image_path}' does not exist.")

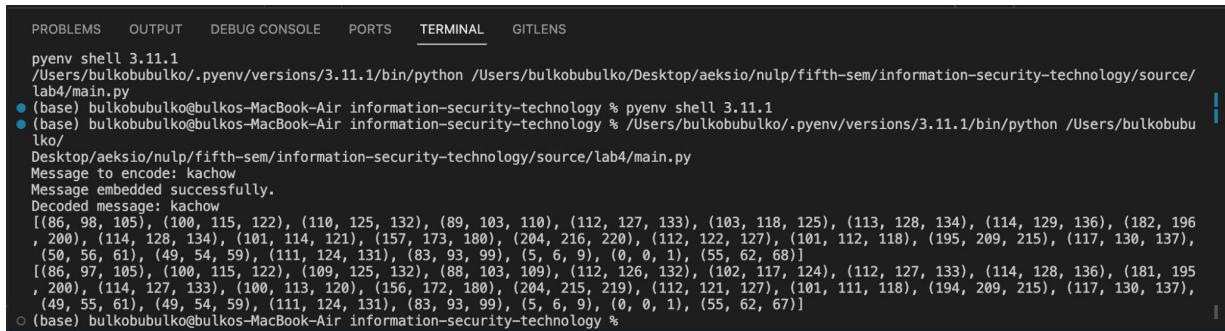
        print("Message to encode: " + message)
        encode(image_path, message, output_path)
        decoded_message = decode(output_path)
        print("Decoded message: " + decoded_message)

        print(get_pixel_values(image_path))
        print(get_pixel_values(output_path))
    except FileNotFoundError as e:
        print(f"Error: {e}")

if __name__ == "__main__":
    main()

```

Приклад виконання зображено на Рис. 1.



```
PROBLEMS OUTPUT DEBUG CONSOLE PORTS TERMINAL GITLENS
pyenv shell 3.11.1
/Users/bulkobubulko/.pyenv/versions/3.11.1/bin/python /Users/bulkobubulko/Desktop/aeksio/nulp/fifth-sem/information-security-technology/source/lab4/main.py
(base) bulkobubulko@bulkos-MacBook-Air information-security-technology % pyenv shell 3.11.1
(base) bulkobubulko@bulkos-MacBook-Air information-security-technology % /Users/bulkobubulko/.pyenv/versions/3.11.1/bin/python /Users/bulkobubulko/Desktop/aeksio/nulp/fifth-sem/information-security-technology/source/lab4/main.py
Message to encode: kachow
Message embedded successfully.
Decoded message: kachow
[(86, 98, 105), (100, 115, 122), (110, 125, 132), (89, 103, 110), (112, 127, 133), (103, 118, 125), (113, 128, 134), (114, 129, 136), (182, 196, 200), (114, 128, 134), (101, 114, 121), (157, 173, 180), (204, 216, 220), (112, 122, 127), (101, 112, 118), (195, 209, 215), (117, 130, 137), (50, 56, 61), (49, 54, 59), (111, 124, 131), (83, 93, 99), (5, 6, 9), (0, 0, 1), (55, 62, 68)]
[(86, 97, 105), (100, 115, 122), (109, 125, 132), (88, 103, 109), (112, 126, 132), (102, 117, 124), (112, 127, 133), (114, 128, 136), (181, 195, 200), (114, 127, 133), (100, 113, 120), (156, 172, 180), (204, 215, 219), (112, 121, 127), (101, 111, 118), (194, 209, 215), (117, 130, 137), (49, 55, 61), (49, 54, 59), (111, 124, 131), (83, 93, 99), (5, 6, 9), (0, 0, 1), (55, 62, 67)]
(base) bulkobubulko@bulkos-MacBook-Air information-security-technology %
```

Рис. 1. Приклад виконання

Посилання на GitHub-репозиторій: <https://github.com/bulkobubulko/nulp-ist>

Висновок: під час виконання лабораторної роботи було вивчено опрацювання (шифрування та дешифрування) файлів з прихованими інформаційними повідомленнями.