MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

NATIONAL TECHNICAL UNIVERSITY

"KHARKIV POLYTECHNICAL INSTITUTE"

Department of Computer Engineering and Programming

«Software Means of Information Protection »

*Laboratory work report No 8*

*Topic: «* *Steganography transformations»*

Student:

Group. KH 919 i.e.

Ayoub EL-HADDADI

Verified by:

Lecturer Viktor CHELAK

Kharkiv – 2022

***Purpose of work:***

Create a program to hide a message in an image file using steganography transformations.

***Individual task:***

• Program to hide a message in an image file.

• Program to extract a hidden message from a file.

Variant 8: Block concealment method.

***Algorithm of the program:***

The program has three functionalities:

1. Hide data inside image file.
2. Extract data from modified image using key generated during hiding.
3. Extract data using original image.

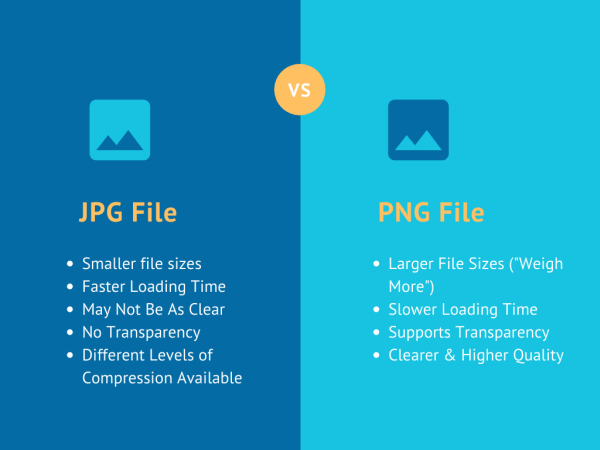
*#hiding data in image:*

Using Bitmap class we can get an instance of existing image, with this instance we can get image pixels, width and height. First step after creating instance of image is to get binary string representation of message to hide and check if it fits the image by calculating how many blocks data needs and maximum blocks in image. Next, we go through each block inside loops and choose a pixel randomly, we add pixel coordinates in an initialized list (which is the key).

For selected pixel we change 2 lowest bits of bytes responsible for red and blue colors. Therefore each pixel in each block takes 4 bits from message.

After editing desired pixels and generating the key we save the modified instance of bitmap in another image in PNG format.

*Note: JPEG and PNG both are a type of image format to store images. JPEG uses lossy compression algorithm and image may lost some of its data whereas PNG uses lossless compression algorithm and no image data loss is present in PNG format.*



*Figure 1 – JPEG vs PNG*

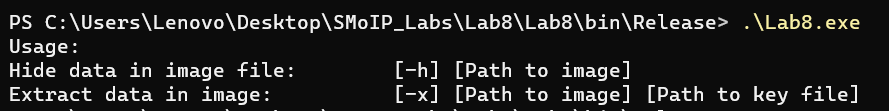
*#Extract hidden message in image using key:*

This functionality takes two parameters, path to image which have a hidden message and path to key generated from hiding this message. And, using a bitmap instance of this image we use coordinates in key and extract 2 lowest bits in each red and blue byte of this pixels. Next we convert this binary string to array of bytes and write results in path selected by user.

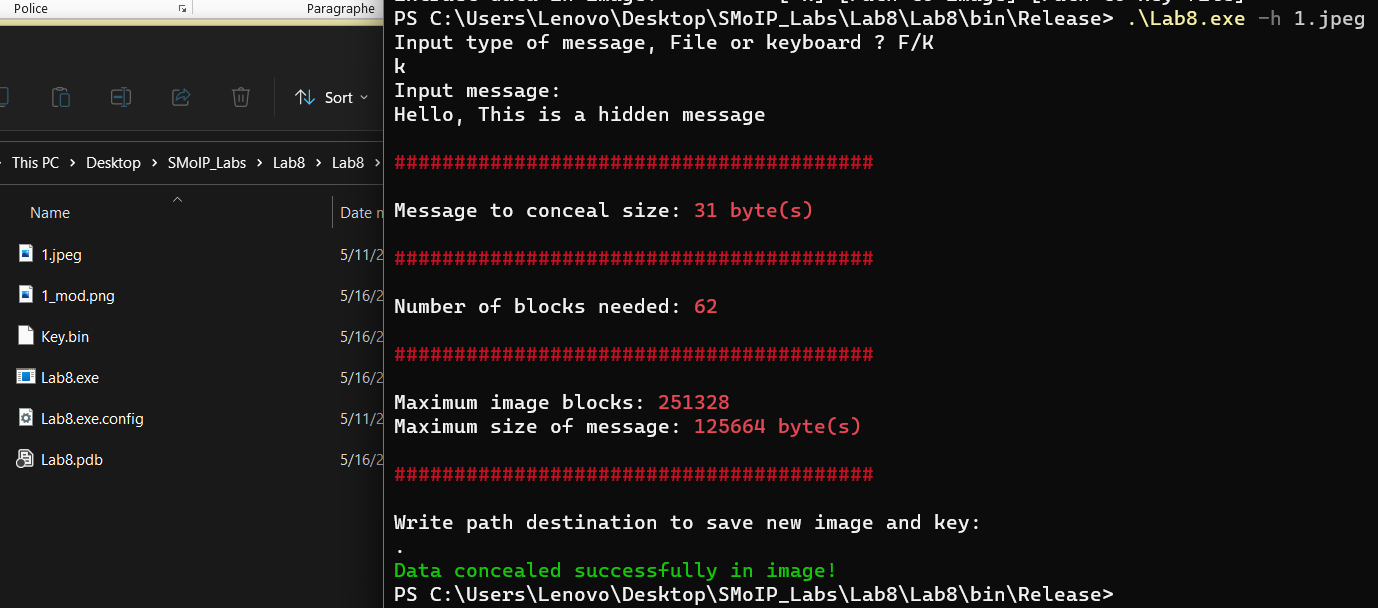
*#Extract hidden message in image with original image:*

To extract message from image with help of original image we just compare pixels with each other (same coordinates) and if they are not the same we take lowest 2 bits in red and blue bytes. But, there is a problem that a pixel may be not changed because 2 lowest bits are the same with bits that we want to change to a solution (if it is a solution) for this problem is during hiding message we add an if statement that check if red and blue bytes changed if not we increase (or decrease) byte responsible for green color. There for in extraction we compare two pixels even if red or blue are equal green will not be equal. So, we take 2 lowest bits of red and blue even if they are the same with original image.

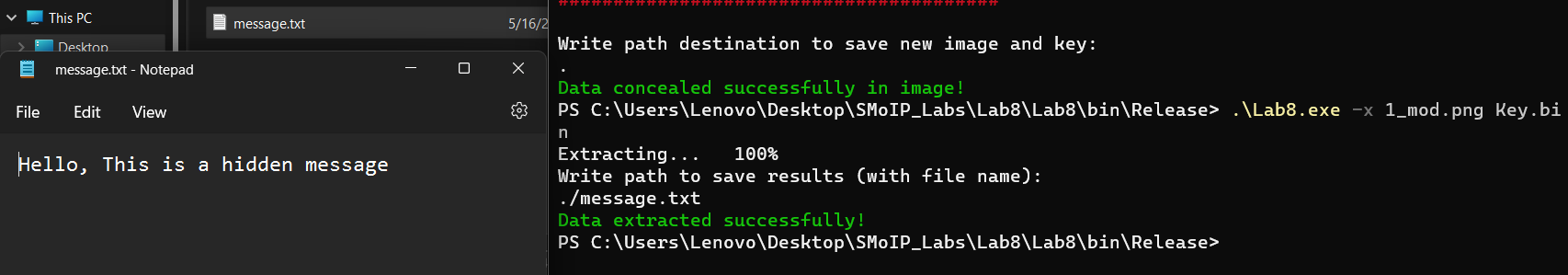
***Screenshots of the program***:



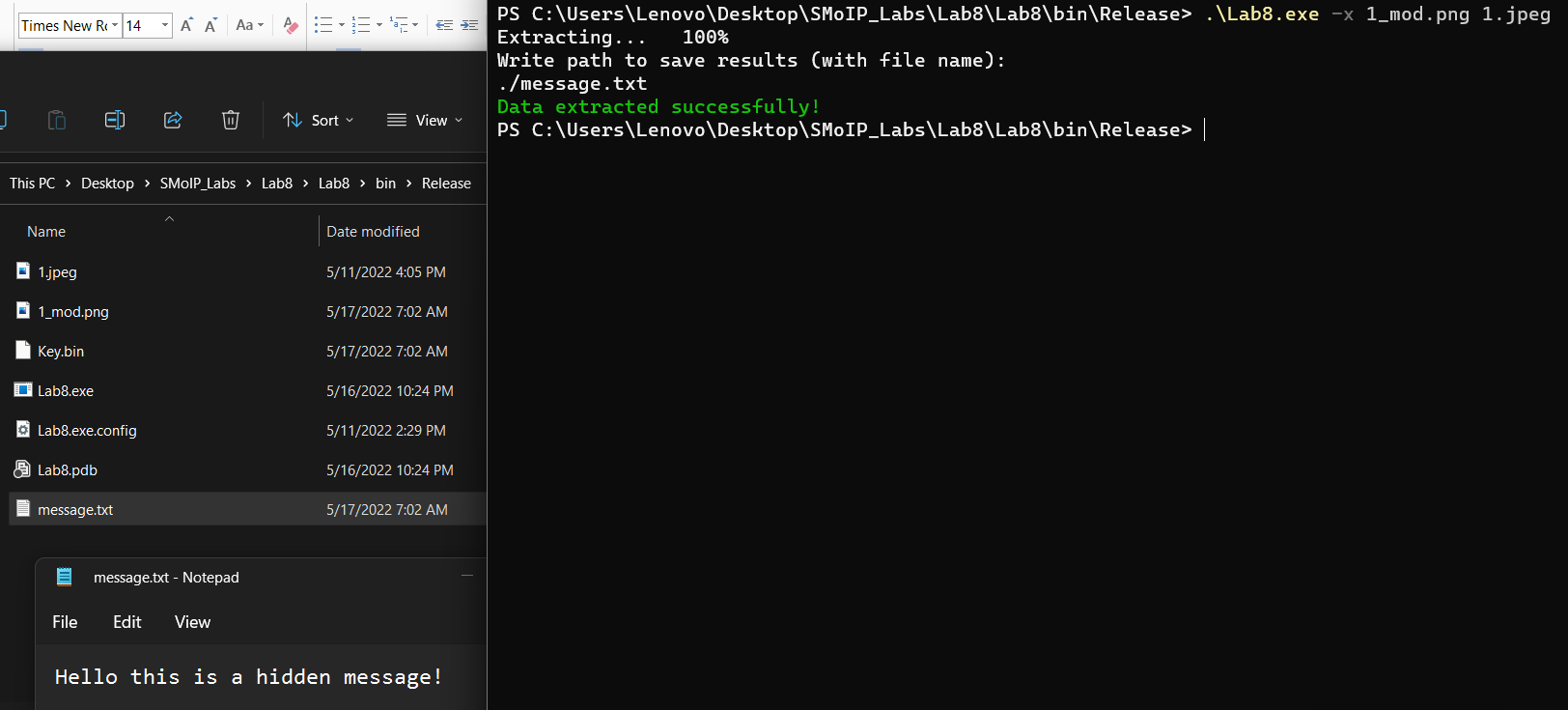
*Figure 2 – Usage of the program*

**

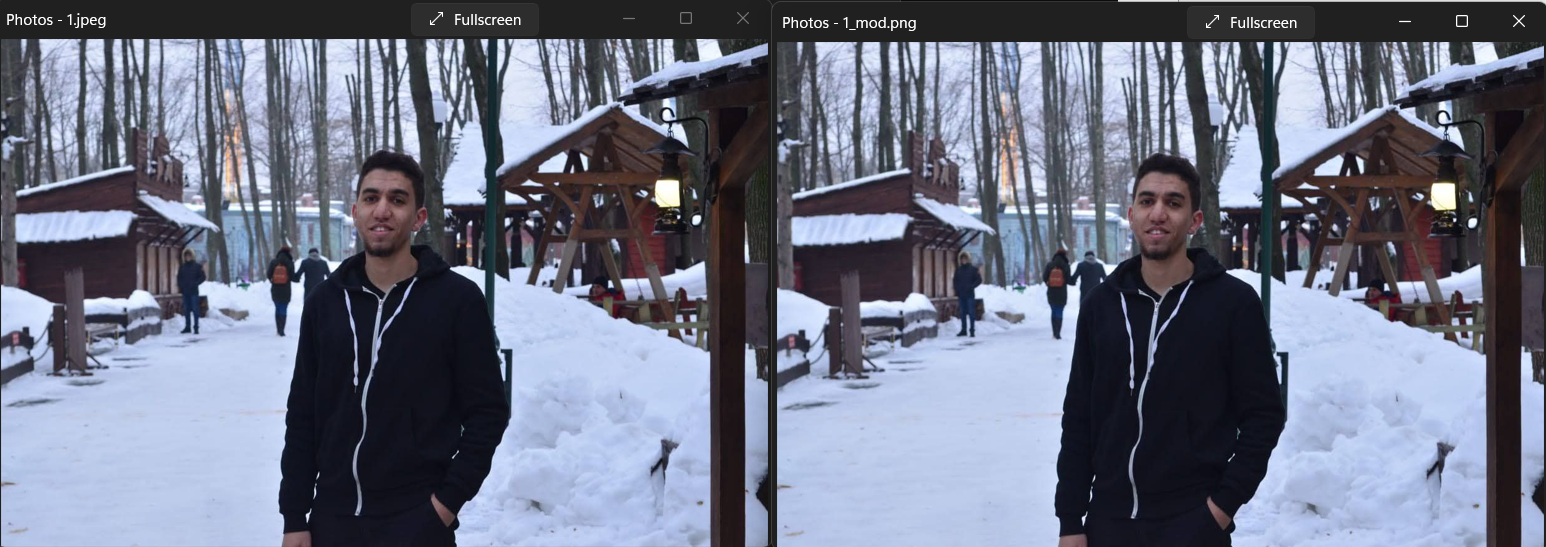
*Figure 3 – Hide 31 bytes message in image*

**

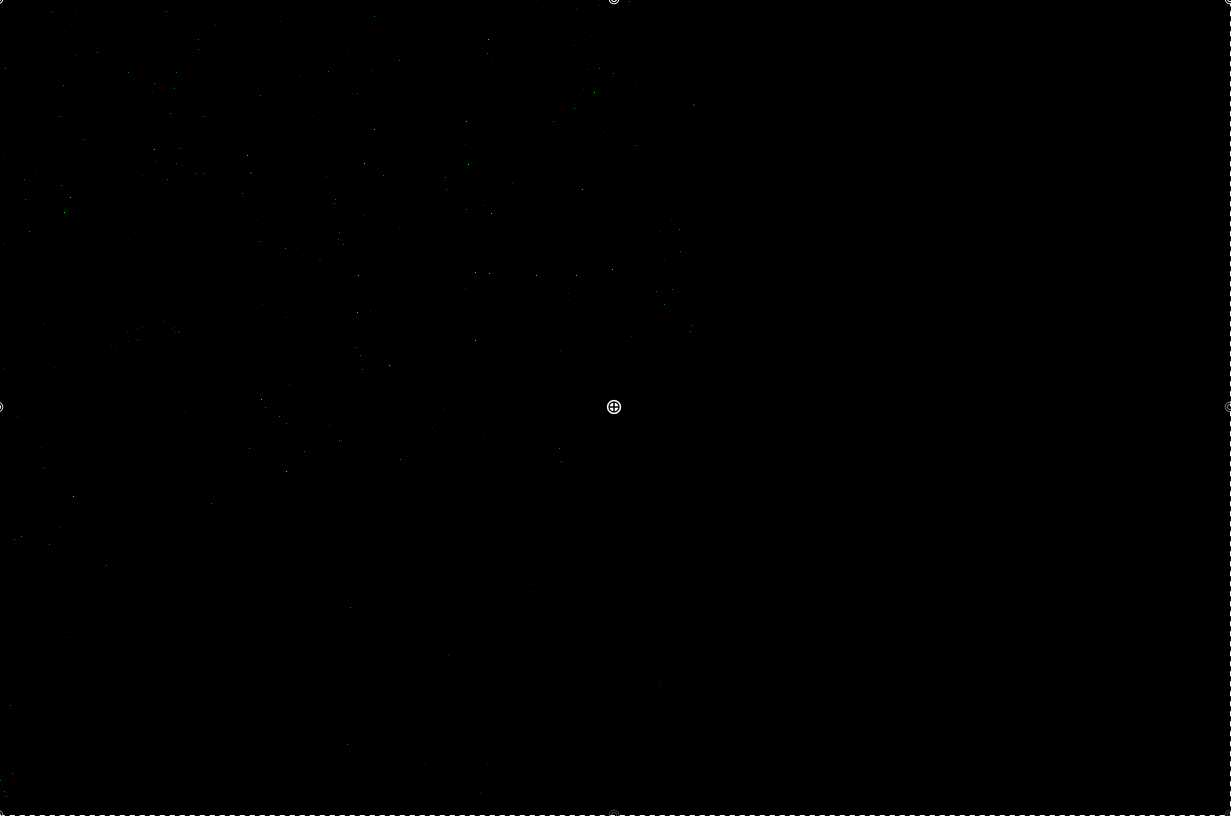
*Figure 4 – Extract message from image using key*

****

*Figure 5 – Extract message using original image*

****

*Figure 6 – Original image – Modified image*

****

*Figure 7 – Difference between image that contains my CV and original*

**Source Code:**

[*https://github.com/Elh-Ayoub/SMoIP\_Labs/tree/main/Lab8*](https://github.com/Elh-Ayoub/SMoIP_Labs/tree/main/Lab8)

**Conclusions:**

For this laboratory work, I have gained principles of developing a program for hiding and extraction from image file using bitmap class, and gained knowledge about difference between image formats.