Information hiding via invisible watermarking (text)

Image processing phase 1

HABIBA AMR 2020/08121

Information hiding via watermarking

Hiding text

# Purpose

|  |  |
| --- | --- |
|  | Hiding information in an image using watermarking can be performed using steganography which is a method of hiding secret data, by embedding it into an audio, video, image, or text file. The purpose of steganography is covert communication to hide a message from a third party. It is one of the methods employed to protect secret or sensitive data from malicious attacks. It’s also useful to fingerprint a file and only the receiver can decode it, and the reason is to help us know where it came from, which can be used for copyright reasons or to trace who’s been distributing illegal material. For example, by using steganography we can add a watermark to an image in the form of a text and repeat it in the whole image, so, if this image is leaked on the internet, and then we find an invisible watermark on it, it means that the person who leaked it is not the real owner, and we can find out where it came from by extracting the watermark. It’s also sometimes used with AI by attackers who develop AI implementations that have even been able to modify steganographic techniques so that attacks can’t be easily discovered. Attackers sometimes use it to hide a document, video, or photograph inside another file that seems legit. For my project I’ll be using steganography to hide a text in an image as an invisible watermark and will also explain how to decode it so that the receiver can read the hidden text in the image. |

# Methods and processes

This is an explanation of the methods used and the processes included:

* The method used to hide the text in the image is steganography in image processing.
* Process 1: Encoding the text into the image.
* Process 2: Decoding the image to be able to read the text encoded in it.

# Process

|  |  |
| --- | --- |
|  | The process starts with the input which is the image, a colored or a gray-scale image can be used, after choosing a suitable input image we get to the next step in our process which is pre-processing, the image contains pixels and each pixel contains a value, we will consider a digital image as a set of values, each value describing a color. in a grayscale image for example, we usually have a set of values that range from 0 to 255, where 0 represents black, 255 represents white, and the values in between represent different levels of gray, these values are arranged in a 2D, when the value is closer to 0 the shade of gray is darker, and when it's closer to 255 it gets lighter. Then we transform all the values in all the pixels into binary values, for example a pixel with the value 104 which equals to 01101000 in binary, and we do that to all the pixels in our image, then we end up with the same image, but the values are now in binary, and this image will be represented as an array that contains the values of the pixels in binary. Then we move onto the next step which is our process, and this is when we start hiding the text into the image, and this is achieved by changing the least significant bit in each byte (8bits) for each pixel after translating our text from ASCII to binary (and the reason why we change the LSB is because the effect on the image will be barely noticeable but if we change the MSB the effect will be visible and easier to detect), for example if the text we want to hide is “hi”, we change the text from ASCII to binary, so the “h” will be 01101000 and the “i” will be 01101001, then we take the first bit (MSB most significant bit) form our text and change the LSB (Least Significant Bit) from the first pixel in our image to be 0 which is the first bit from our text and continue this process until we finish hiding the bits of our text, then we save the image after modification and display it which will not look really that different from the original and with that we get the last step which is the output image after hiding the text. And this is for the encoding process.  For the decoding process we do the same thing except the process step. After transforming the image pixels into binary we then extract the LSB of every value in each pixel (and we can use a function that sets the limit for the extracted LSBs according to the size of our text, which we can also save in the image to help set the limit when extracting), then we form groups of 8s of these bits to form bytes which will be the bytes of our text which will then be transformed from binary to ASCII to finally give us the text that was hidden in the image, which will finally give us the output for the decoding process which is the text that was hidden in the image. |

# Block diagram

For the encoding process:

Input

Pre-processing

Process

Output

The output image after hiding the text.

The input image which can be colored or a gray-scale image.

We transform all the values in all the pixels into binary values, for example a pixel with the value 104 which equals to 01101000 in binary, and we do that to all the pixels in our image, then we end up with the same image, but the values are now in binary, and this image will be represented as an array that contains the values of the pixels in binary.

This is when we start hiding the text into the image, and this is achieved by changing the least significant bit in each byte (8bits) for each pixel after translating our text from ASCII to binary, for example if the text we want to hide is “hi”, we change the text from ASCII to binary, so the “h” will be 01101000 and the “i” will be 01101001, then we take the first bit (MSB most significant bit) form our text and change the LSB (Least Significant Bit) from the first pixel in our image to be 0 which is the first bit from our text and continue this process until we finish hiding the bits of our text, then we save the image after modification.

For the decoding process:

The output for the decoding process which is the text that was hidden in the image.

The input image which can be colored or a gray-scale image. But this is the image after hiding text in it, which is the steganographic image.

We transform all the values in all the pixels into binary values, for example a pixel with the value 104 which equals to 01101000 in binary, and we do that to all the pixels in our image, then we end up with the same image, but the values are now in binary, and this image will be represented as an array that contains the values of the pixels in binary.

After transforming the image pixels into binary we then extract the LSB of every value in each pixel (and we can use a function that sets the limit for the extracted LSBs according to the size of our text, which we can also save in the image to help set the limit when extracting), then we form groups of 8s of these bits to form bytes which will be the bytes of our text which will then be transformed from binary to ASCII.

Input

Pre-processing

Process

Output

# References

1. [Image Steganography in Cryptography - GeeksforGeeks](https://www.geeksforgeeks.org/image-steganography-in-cryptography/)
2. [What Is Steganography | Cybersecurity | CompTIA](https://www.comptia.org/blog/what-is-steganography#:~:text=The%20purpose%20of%20steganography%20is%20to%20conceal%20and,doesn%E2%80%99t%20involve%20scrambling%20data%20or%20using%20a%20key.)
3. [Hackers’ latest weapon: Steganography (computer.org)](https://publications.computer.org/computer-magazine/2018/11/15/how-steganography-works/)
4. [FBI — Forensic Science Communications - July 2004](https://archives.fbi.gov/archives/about-us/lab/forensic-science-communications/fsc/july2004/research/2004_03_research01.htm)
5. [Hiding Data Inside an Image Using Python : 7 Steps - Instructables](https://www.instructables.com/Hiding-Data-Inside-an-Image-Using-Python/)