Steganography
Computer Security IV1013

Axel Månson Lokrantz

2023-06-06

Question 1: It is up to you to choose if you want to use audio or image format for the target file. In both cases, there are many different standards to choose from. For instance, for image files there are JPEG, PNG, TIFF, and GIF, while for audio formats there are CD, WAV, MP3, Ogg Vorbis, and FLAC. Which one did you choose? Explain your choice. What alternatives did you consider, and why did you prefer your choice?

Answer: I chose to work with images instead audio because images provide visualization, making it easier to inspect the output file for visual artifacts. In contrast, with audio files, detecting changes requires listening carefully to audio samples to identify any alterations audibly. As for the format, I decided to use PNG because of the compression it utilizes. PNG uses lossless compression which means it preserves the exact pixel value of the original image, this type of compression is vital for steganography because manipulation of pixels during compression could potentially ruin the concealed data. Formats such as JPG and GIF which use lossy compression were therefor not considered.

Question 2: You need to store metadata in the target file as well. At least, the length of the data file needs to be represented somehow. Describe and explain your solution.

Answer: I decided to store the length of the data file at the very beginning of the image. This was a convenient solution because an int is 4 bytes of data, meaning the first 32 bits that are extracted from the image will represent the length. With the length at hand, I could extract the subsequent bits from the pixels.

Question 3: Clearly there is an upper limit for how large n you can use without noticeable effect in the data file. Do some experiments and report your findings.

Answer: The figure below is an experiment I did with an image of the size of 20x20 pixels. For small values of n the data is well concealed and the color shifts are hard to detect. Even more so in images of larger dimensions. There is no universal right answer of what n to use, factors like size, color and contrast all come into play when concealing your data inside the image. The amount of data that can be concealed depends on the dimensions of the image. The larger n is, the more data you can store. Let's consider an image with a size of one gigapixel. When the value of n is set to 1, you can encode 1 million bits of data within that image. However, if n is increased to 4, you can encode four times that amount. However, increasing the value of n comes with a trade-off: the image's colors start to shift, which could potentially be detected.

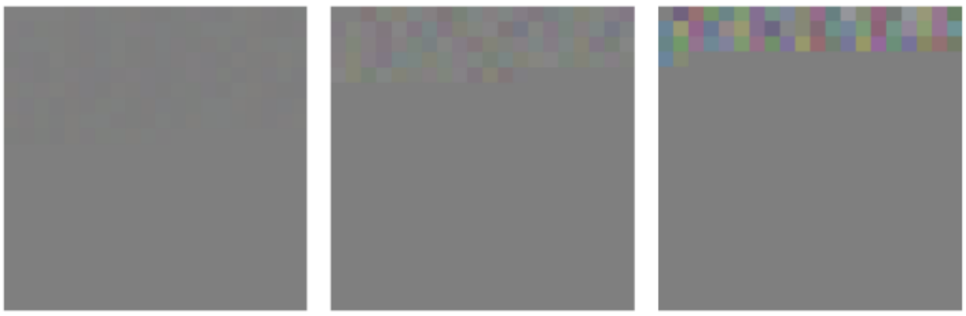

Figure 1: Left most: n = 2. Center: n = 4. Right: n = 6.

Question 4: Obviously there are many different values for c' to chose from when storing a data value v. Any c' such that c' $mod 2^n = v$ would do. Discuss different strategies for choosing c'.

The technique i used is similar to least significant bit substitution. I iterate through all the possible values for c' and compare them to c. Whichever value that is closest to c is selected as c' as seen below.

```
private  byte getcPrime(byte pixelByte, int v, int n) {

        int c = Byte.toUnsignedInt(pixelByte);
        int cPrime = Integer.MAX_VALUE;
        for (int i = 0; i <= MAX_COLOR_VALUE; i++) {
            int tmp = i;
            if((tmp % Math.pow(2, n) == v)){
                if(Math.abs(c - tmp) < Math.abs(c - cPrime))
                    cPrime = tmp;
            }
        }
        return (byte) cPrime;
    }
```

However, there are other techniques that are better from a security standpoint. Instead of always choosing the least significant bits a predetermined pseudo-random sequence can be used. The altered bits are then chosen at "random" by the sequence which is generated through a key that both the sender and receiver are aware of. Additionally, another technique is called adaptive steganography where the choice of c' is determined from the neighboring pixels, which allows for a more seamless integration of the data. The aim for such method is to minimize visible anomalies that may arise during encoding.