# Introduction

Abstract: Digital steganography is the practice of using some digital formats, such as image files or sound files, as a cover for hiding messages, without altering the original files in terms of human perception. This method of concealing data may use any computer file type as a container in which we can embed the same or another digital file type. For example, we can hide: a plain text file into an image file, a picture into a sound recording or an image into a text file. However, it is not always possible to do this due to the sizes of the files, so hiding some text into an image file may be far easier than the other way around because, generally, an image file is bigger in size than a text file, but more important, a text file doesn’t contain redundant information, in comparison with a picture. For this reason, this paper will be a description of how to embed secret text messages in image files. This computer medium is generally large and redundant enough that we can alter it, by embedding the secret information inside, but to the human eye it will appear the same. This is a clever way to send sensitive information because, while cryptography ensures the message is not readable, steganography does not reveal the existence of the message at all. For a better security, cryptography will also be used to encrypt the messages before embedding them into the cover.

Key-Words: image steganography, least significant bit, digital steganography, data hiding,

cryptography, steganalysis

1 Introduction

For as long as we have known ourselves as humans, besides other problems such as food, water or shelter, we have also had the difficulty of communicating sensitive information. While this dilemma existed in our minds since the birth of humankind, rudimentary solutions started to be implemented back in the ancient times. Nowadays, in an era of information technology, where virtually anybody owns some kind of device connected to the internet, which provides access to lots of data, digital steganography is the modern way of concealing messages or other delicate information. As opposed to ancient steganography techniques, which were implemented in a physical manner, using special ink, writing the message in an unobservable area of a letter or using certain rules for extracting the concealed information from an otherwise normal looking text, the modern implementations makes use of the digital form in which information circulates now.

The term steganography comes from the Latin word “steganographia” which originated from the Greek words “steganos”, meaning concealed and “graphia”, meaning writing, so steganography describes, generally, any kind of concealed writing.

Digital steganography is the process of embedding sensitive pieces of data into another computer file in such a manner that human perception cannot detect the difference between the original file and the one containing added information. For the purpose of this paper, the confidential data will be referred to as the secret message and the container which embeds it will be called the cover file. Generally, both the secret message and the cover file can be of any computer file type. That means a text file can be hidden into an image file, but it can be done the other way around too, although that would be harder because of the generic redundancy and size difference between the two types.

Images are appropriate mediums for the steganographic process due to the large size and high redundancy.

For the purpose of this paper, images will be used as the cover file type and plain text as the secret message type. More specific, the BMP (Bitmap Image File) image format will be used as the cover file.

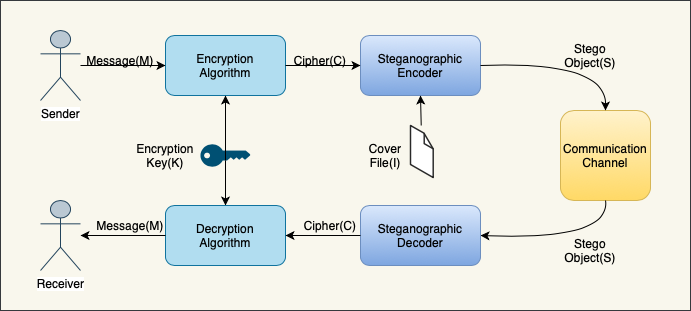
 Additionally, encryption will be used in order to ensure better security for the message, by encrypting it before embedding. Cryptography is different than steganography because, while encrypting a message ensures its transformation into a form that an eavesdropper would not understand, steganography tries to hide the existence of a hidden message at all.

Figure 1 - Steganographic schema

Other similar processes tightly coupled with steganography are fingerprinting and digital watermarking. A fingerprinting algorithm is used to generate a unique mark for a piece of data and embed it into that specific file. This is very useful when you want to supply some files and protect them from ongoing distribution. Watermarking also embeds a mark of the files with the purpose of signifying ownership. As opposed to steganography, in fingerprinting and watermarking the existence of the embedded data is publicly known, whereas steganography tries to completely hide that there is any information hidden inside. While a successful attack concerning a watermarking or fingerprinting algorithm consists of removing the watermark or fingerprint, basically removing the ownership protection, an attack on a steganographic system should detect and eventually extract the hidden data.

2 LSB Steganography Technique

LSB or Least Significant Bit steganography is a technique which implies using the redundancy of the information as a place to store the data to be hidden. This is the reason multimedia files are so popular with steganography. They contain a fair amount of redundant information, whether we discuss about image, audio or video files, which makes the perfect home for secret messages. On the other hand, from a statistical point of view, if the message is embedded directly, as it is, into a cover file, it may be easily discovered, because altering the original file this way changes its statistical properties, if the secret message is large enough.

This is another good reason to also use cryptography when applying this kind of technique. Besides securing the content of the message, through encryption, the original message is transformed in a more random looking form, that of a ciphertext. This way, the statistical impact of the embedding process is drastically reduced because the randomness of the least significant bits is not altered heavily.

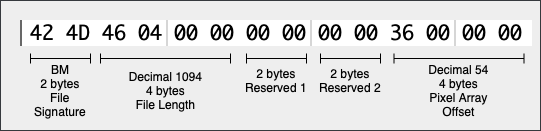
 In my opinion, an alternative solution to using cryptography would be the drastic limitation of the size of the secret message. For example, in a scenario in which LSB image steganography is used to embed secret messages in photos sent over the network, through a mobile application, considering the fact that two pixels are needed to hide one character, I would define the steganographic capacity of the cover file having width and height as:

Figure 2 - BMP File Header Example

This makes sure that the number of modified pixels in the cover image is at maximum 0,5% of the total number of pixels, which statistically should be impossible to detect.

To further describe the steganographic process, the structure of the used files should be understood. The next section details the data organization in the files used for the purpose of this article.

3 Steganographic Methodology

This chapter details how to embed a secret text message into a BMP (Bitmap Image File) cover file, using the least significant bit technique described before. In order to understand the process, the data stored in the files we are working with should be detailed. Those files are the secret message and the cover file.

The embedded secret message is a sentence or piece of plain text, which will be hidden and is represented inside a text file as a vector of ASCII codes, one for each character. For example, the code for letter “A” is 65 in decimal format, 41 in hexadecimal format or 0100 0001 in binary format. Therefore, if we want to hide the letter “A” into an image we have to insert somewhere in the image the binary sequence 0100 0001.

The cover file format used for the purpose of this paper is BMP (Bitmap Image File), which, as the name suggests, is a raster image file, meaning that the image data is stored inside the file as a matrix of pixels and their color values. To better understand the steganographic process, the BMP file structure should be detailed. For the description of the methodology it is necessary to understand the two major components of the file: the header and the pixel array. The file begins with its header, which is 14 bytes long and contains data about the file, in order, as explained below:

* the first 2 bytes of the file are representing the file signature, which in this case consists of 2 letters: usually BM
* the next 4 bytes represent the size of the entire file
* the next 4 bytes are reserved for different uses
* The last 4 bytes represent the offset in the file at which the pixel array starts

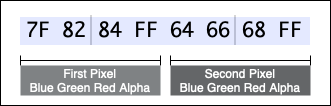
 The above example presents a real BMP image file header which was described before. It’s important to keep in mind the offset at which the pixel array starts, which in this example is 54. Therefore, all the bytes beyond position 54 represent the pixels and their color and transparency. The format used in this paper follows the RGBA color model, so each pixel is defined by 4 bytes, one for each channel, which means every dot used to represent the image is defined by four values: the amount of red, green, blue and alpha.

Figure 3 - Pixel array example

In Figure 3 is presented an example of the first two pixels in the array of a BMP image file. The values seen in the example below are the values of the blue, green, red and alpha channel, respectively. In binary format, the two pixels data is:

01111111 10000010 10000100 11111111

01100100 01100110 01101000 11111111

The value of each channel, red green, blue and alpha, using 8 bits, is in the 0-255 range. If the rightmost bit is altered, the value increases or decreases by 1. The intensity difference of that particular color of a particular pixel is 1 at most, which is basically invisible to the human eye.

It is important to know how pixel information is stored to understand how the steganographic technique described in this article works.

Considering the concrete example from above, the process of embedding the secret text message into the cover file follows the next steps:

1. Encrypt the content of the text file. For this step asymmetrical encryption will be used so each party trying to communicate needs a set of public/private keys, which will also be used to sign the cover file, in order to protect it from tampering of any kind.
2. Extraction of the pixel array data from the cover file, in order to process it.
3. Modification of the least significant bits of the pixels to be the same as the ciphertext of the secret message.
4. Signing the altered image to protect it from tampering. If an attacker would alter some of the content of the image or the image gets compressed, it would be known, because the cover file signature will not match anymore.

If we take the above example the secret message “A” would first be encrypted using the recipients public key. The result is a 1024-bit long cipher text as seen bellow:

“A” – 01000001 (binary format)

Ciphertext - 00010110 11100100 01110001 01100110 01011111 …

Original first pixels - 01111111 10000010 10000100 11111111 01100100 01100110 01101000 11111111

Altered first pixels - 01111110 10000010 10000100 11111111 01100100 01100111 01101001 11111110

In the above example it was shown how to embed the first byte of the ciphertext into the pixel data of the cover image. Notice that two pixels are needed to embed a byte of ciphertext, but the actually changed values are corresponding to four out of eight bytes of the pixels, because the other four bytes happen to have the least significant bits the same as our encrypted message.

Figure 4 - Cover image vs. stego-object

 Another important security aspect to keep in mind, besides the statistical analysis which can be done on the stego-object is compression. Usually, compression algorithms used for high redundancy file formats try to reduce this redundancy to lower the file size, keeping the content almost as it was before. If a lossy compression algorithm is used on a stego-object, it will remove all the secret information embedded into it.

Also, it is very important to discard the original cover file, because if an attacker somehow gets both the cover image and the stego-object containing the secret message, if a subtraction of the two pixel arrays is performed, the result would be the secret message fully disclosed.

In Figure 4, we can see that both the initial cover image and the stego-object look pretty much the same, but if the difference of the pixels in these images is calculated, it can be easily observed that the two are not the same and the difference between them, if decrypted, will disclose the precious and sensitive information inside.

4 Conclusion

In conclusion, steganography is a smart way of hiding sensitive information inside different carrier files. While there are lots of different techniques of performing this task, some more straightforward and some more complicated, the steganographic process should take into consideration the steganalysis methods in which the existence of the secret message might be exposed, or even worse, retrieved from the stego-object. Better statistical approaches can be used to determine the steganographic capacity of a cover file, so steganalysis can’t find the presence of the hidden message using statistical methods.

My approach of performing digital image steganography combines it with cryptographic algorithms to secure the sensitive information, as well as minifying the statistical impact on the original data, which should make a steganalysis process much harder.

Also, the ciphertext should be as evenly as possible distributed through the cover file, for a better security over statistical methods of examination.

# General presentation

# Application implementation

# Application usage

# Conclusions and recommendations