**Steganography Analysis**

Submitted by:

Vishnutej Mylavarapu

Naman Jain

1. Introduction

***Steganography:***

Steganography is the process of concealing a file, message, image, audio, video with in other another file, image, audio or video. Examples include hidden text in a web page, hidden text in an image, hidden image in an image. We chose Image Steganography for our analysis and performed text in image and image in image Steganography. A user who wants to send a secret message to some other user embeds the secret in an image and sends it. The other user who knows which algorithm has been used to embed the message and the fact that the image has a secret message reverses the embedding process to obtain the actual secret message. The process is similar to Cryptography. The advantage of Steganography over Cryptography is that it is not obvious if an image contains a secret message as the image might look exactly similar or almost like the original image. The image can also be sent out in the open.

We can select images with large sizes to store large volumes of data. Storing small amount of data in a large file result in fewer distortions of the original image.

In order to provide more security to the secret information, we could use cryptography to encrypt the text before embedding the cipher in the image. The recipient should first get the cipher from image and then decrypt the message using the secret key.

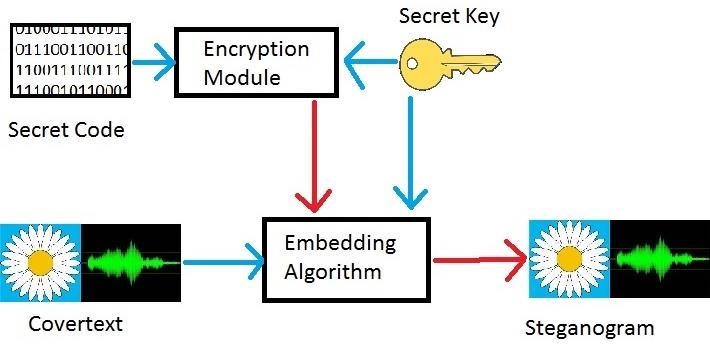


Image reference: <https://null-byte.wonderhowto.com/how-to/introduction-steganography-its-uses-0155310/>

There are several other uses of Steganography. Some of them are:

Steganography can be used by hackers to distribute malware or in attacks such as phishing. Hackers can also use Steganography to send commands to systems that already have malware running. It can be used to control botnets in order to obfuscate any commands that are being sent.

Steganography can also be used to watermark documents. Digital watermarking is used to hide a logo, text or an image to at least partially authenticate the source of the document. An example could be, before a photographer publishes an image in a website, he embeds a signature of his own in the image. This could come in handy if he must prove the ownership of the image.

***Steganalysis:***

The process of detecting and breaking Steganography. We could detect if an image has been used for steganography by examining the color palette since each color has a unique binary encoding. If Steganography has been used, there would be duplicate encodings of the color.

Even if someone suspects that images are used for Steganography, the False Positive rate and the False Negative rate will be high. Retrieving the secret will need the knowledge of the algorithm that was used to embed the information or brute force all possible algorithms which might not be feasible.

1. Algorithms

There are a variety of algorithms available for storing the required data in the image. The way the data is stored in the image affects the distortions of the image. Choosing a bad algorithm can also result in easy identification of images that have additional information in them.

Some of the widely used algorithms for Steganography are:

* **Least Significant Bit (LSB)**

Least Significant Bit algorithm uses the last bit of the pixel data to store the secret instead of the original picture information. The last bit can be thought of as the ones place of a number. We use the least significant bit as the changes in the least significant bit has the minimal effect on the actual image. This replacement can be extended to other bits of the original image. The more bits replaced, the more distorted the original image is. The more bits replaced, the more bit depth available, and the larger the image, the more data that can be stored in the photo.

**Good LSB:**

Following is the algorithm for good LSB:

1. Read the original image.
2. Get the message from the user and use cryptography algorithms to encrypt it.
3. For this, the user also enters a secret key that is used to encrypt the message.
4. Convert this into bits.
5. Read every byte of the image.
6. Only work on the bytes that have the LSB as 0.
7. Now, as the bytes represent R, G and B components, we can change the LSB of any component. Generally, it is the blue component.
8. For decryption, the user enters the secret key and the message is displayed.

**Bad LSB:**

Bad LSB and good LSB do not have a lot of difference. Yet, as seen in the results, we find that there’s a huge difference in the encrypted image. There are 2 major reasons behind this. Firstly, in Bad LSB, the last bits of each acquired byte are changed whether they are 0 or 1, which is not the case with good LSB. Another big factor is the encryption technique. The program, at the time of encryption and decryption, asks the user to enter a 4-digit encryption key. This numerical key is the length of the bits that the program writes directly from the input image to the output image without changing, so that the secret symmetric code is only known by the concerned parties. Obviously, if a wrong code is entered while decryption, wrong output will be shown.

* DCT

DCT stands for Discrete Cosine Transform. It works by calculating the frequencies of the image and then replacing some of them. DCT algorithms are subtler in the way they manipulate photos and so are harder to detect.

The algorithm follows these steps:

1. Read original image.
2. Read secret message and convert it in binary.
3. The original image is broken into 8×8 block of pixels.
4. Transform successive 8 x 8 pixel blocks of the image into 64 DCT coefficients each.
5. Working from left to right, top to bottom, subtract 128 in each block of pixels.
6. DCT is applied to each block.
7. Each block is compressed through quantization table.
8. Calculate LSB of each DC coefficient and replace with each bit of the secret message.

To get back the secret text, we must reverse these steps.

* Append Bits

This algorithm is similar to LSB but instead of using the Least Significant Bit, this algorithm appends extra bit of the secret information to the existing original image data. This distorts the image quite a bit and it becomes easier to detect Steganography.

* F5
* SSIS

Our initial intention was to analyze how each of the algorithms were distorting the image based on the amount of text data. But we realized that there were either no visible distortions or no obvious distinctions in the distortions from the algorithms. Hence, we decided to use LSB and modify the way we implemented LSB and perform text in image and image in image Steganography.

1. Encryption types

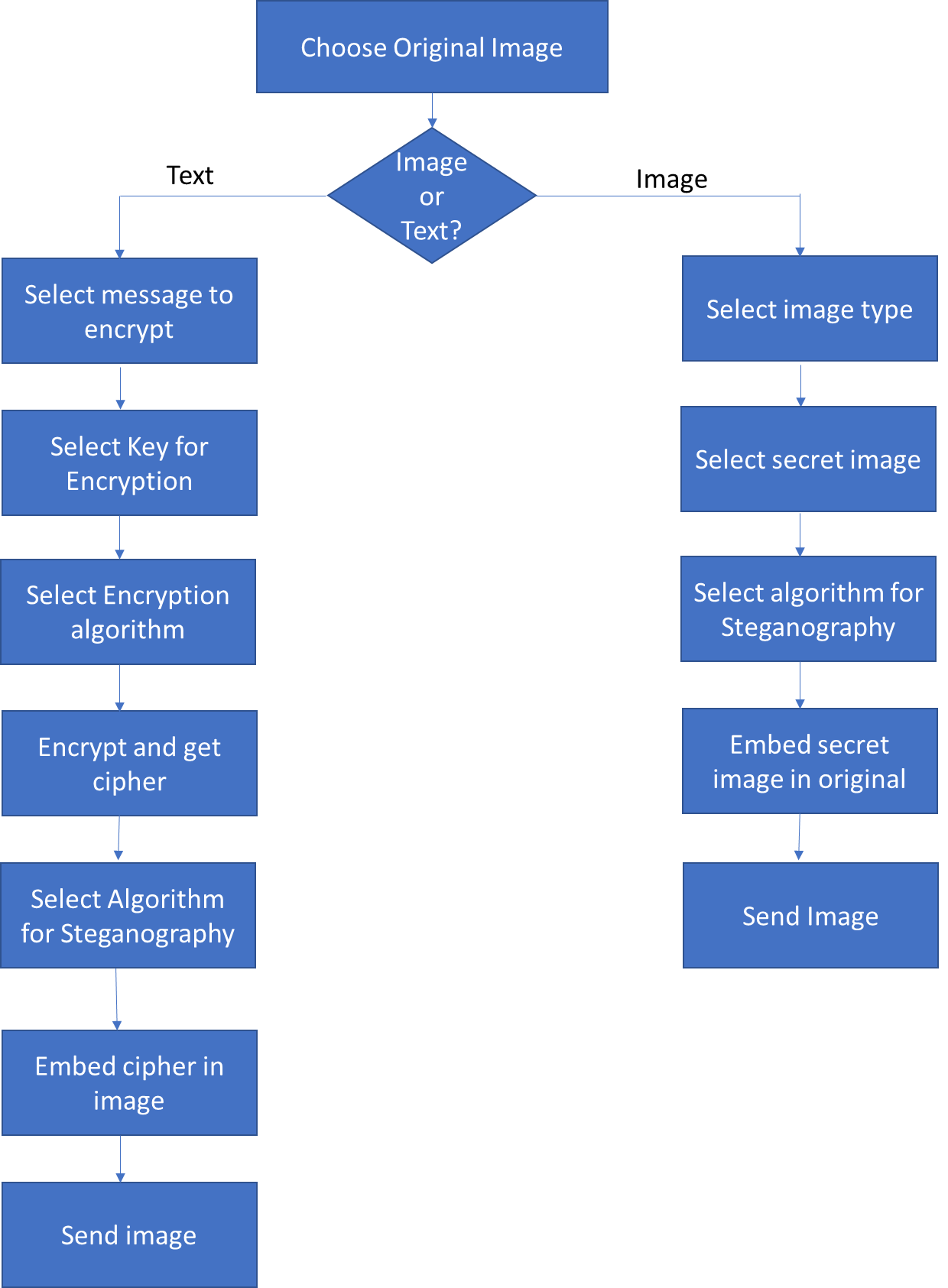
We implemented encryption methods to provide an extra layer of protection for the secret as the image can be sent in the open. The user can choose from a variety of encryption algorithms such as AES, DES, RSA and the recipient has to know which algorithm has been used to encrypt in order to decrypt the secret message.

As the distortion is directly proportional to the data to be stored in the image, we also tried using different sizes of input texts such as 8, characters, 16 characters, 32 characters and 64 characters. As expected 64 characters text distorted the image the most.

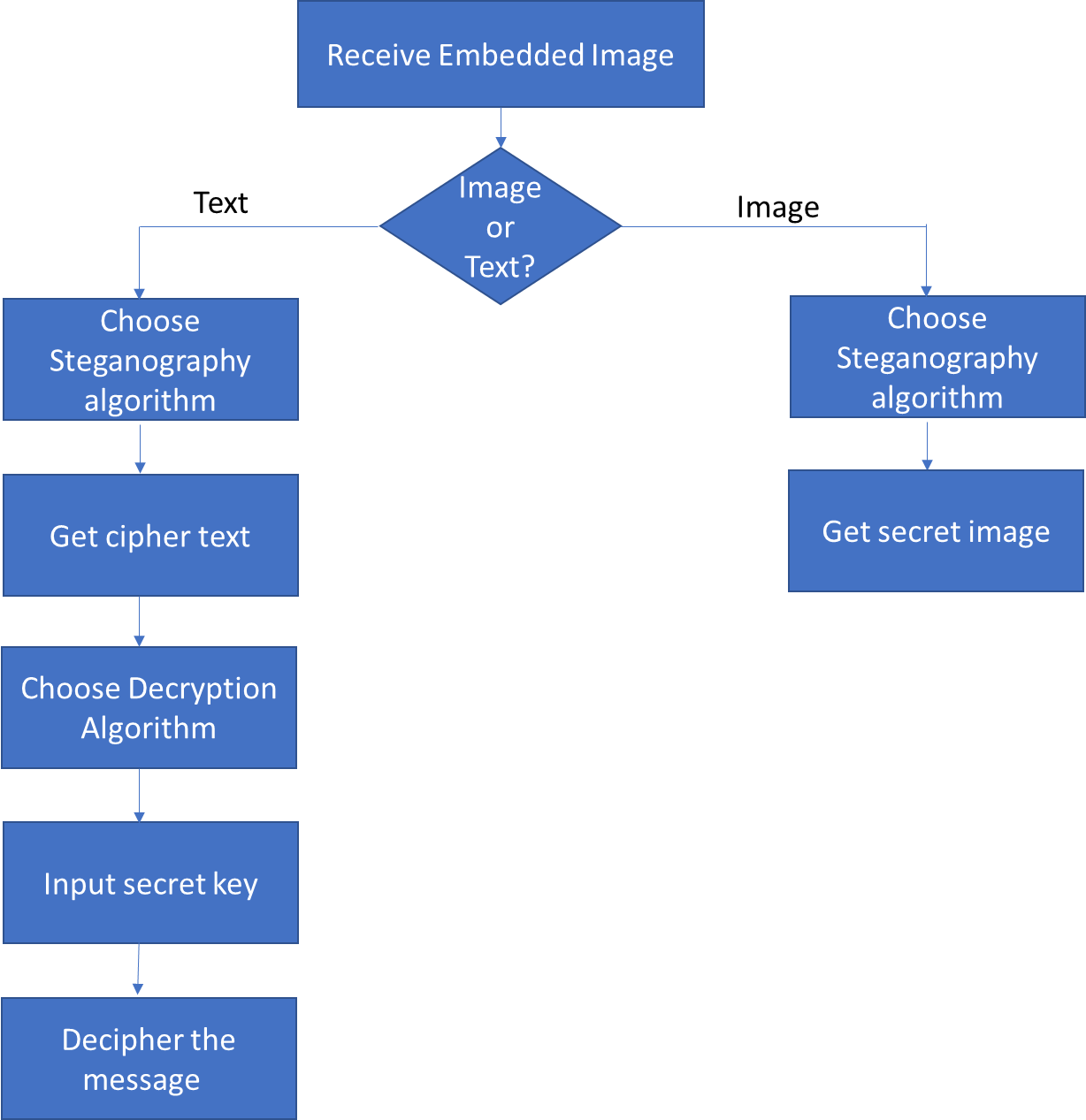
We could also use compression techniques to reduce the length of the secret text. Using compression techniques on images might result in loss of information.

1. Flow chart of the project

Sending:



Receiving:



1. Project Details

As shown in the flowchart we let the users select which kind of steganography they would like to use. There are two options, text in image and image in image. In the case of text in image, we then ask them for inputs regarding the secret message they want to send, the secret key they want to encrypt the secret message with. After encrypting the message, we ask the users to select the type of steganography algorithm to embed the text in the image. We embed the secret text in the image and output the image containing the secret. In the image in image case, we ask user to select the secret image and the steganography algorithm. After embedding the secret image in the original image, we output the image containing both the images. The recipient should reverse the process to obtain the actual message. The recipient must have some information regarding the type of steganography algorithm used, the type of encryption used to be able to get the secret message.

Though we wanted to use various algorithms to check for types of distortions, we were unable to identify any specific patterns that differentiated the various types of Steganography algorithms used, we then thought of varying the secret text sizes to find the amount of distortions. We also tried using online filters in order to identify images that had steganography in them and were able to identify the secret image.

1. Analysis with screenshots and explanations

* Text in Image with Bad LSB

As observed using this algorithm distorts the image a lot and the amount of distortion varies significantly with respect to the secret text.

 Original Image



Image encrypted with shorter text (6-8 letters)



Image encrypted with larger text (A few sentences)

* Text in Image with Good LSB

This is the optimal way of embedding the secret information into the image. The amount of distortion using this algorithm is minimal and we can barely see any difference in the two images here.

 Original Image



Image with embedded text

* Image in Image with LSB

Image in Image stores a large amount of data in the image. So, distortions are visible and using the LSB (or DCT) we can minimize the distortions.

Secret Image:



Cover Image:



3. Resulting image



1. Results summary

Image in image: We observed distortions in the secret image. Using JPEGs and compressing, leads to loss of data. Although we were able to extract the exact string while using text in image, the problem is obvious while using image in image as we were not able to get back the image as it was. Also, the larger the data was, the more time it took to embed the data. This was also the same while retrieving the data. Although, there is a very less difference in the input and the output, it is still not perfect.

Text in Bad LSB: This is the one with the most problems. Because of the encryption technique and the other difference from the Good LSB, there’s a lot change once the text is embedded in the original image. The output image depends on length of the embedded string.

Text in Good LSB: There’s hardly any change. However, working with different images of different sizes and changing the red, blue and green components of each might result in a slight change (not more than a small thin line of a different color).

1. Problems with image types

GIF and 8-bit BMP files employ what is known as *lossless* compression, a scheme that allows the software to exactly reconstruct the original image. JPEG, on the other hand, uses *lossy* compression, which means that the expanded image is very nearly the same as the original but not an exact duplicate. While both methods allow computers to save storage space, lossless compression is much better suited to applications where the integrity of the original information must be maintained, such as steganography. While JPEG can be used for steganography applications, it is more common to embed data in GIF, PNG or BMP files.

1. Pros and Cons of Steganography

*Pros:*

* Sending hidden images and text while avoiding the man in the middle.
* Water marking
* Better than cryptography in the sense that it is not obvious that steganography has been used.
* Variety of media to use for steganography

*Cons:*

* Used for illegal practices.
* Can be broken using brute force attacks depending on encryption techniques.
* Slow

1. Future Work

We could implement filters to identify the secret images inside the original image and depending on red, green, blue filters used and based on which LSB technique was used, we could identify the secret image. There’s a very marginal change in the original image after the secret image is embedded in it but when we use filters, we can easily identify what image has been hidden in the cover image.

Implement methods for detecting steganography. We could use Machine Learning to train models to detect images that are used for Steganography.

1. References

<https://www.garykessler.net/library/steganography.html>

<http://ijact.org/volume3issue4/IJ0340004.pdf>

<https://null-byte.wonderhowto.com/how-to/introduction-steganography-its-uses-0155310/>

1. Code

<https://gitlab.oit.duke.edu/vm73/Steganalysis>