The Australian National University

[[1]](#footnote-1)

Multi Image Steganography

31 May 2017

Javed Ashraf (u6272684, %)

Mohit Reddy Kondreddigari (u6114363, %)

*Abstract* — Summarize your work here with 100~200 words. Briefly discuss the aims, methods, experimental results and key finding of the work.

# Introduction

You can finalize this section with a brief outline of the sections following.

# Related Work

Try to provide what has been done, according to literature survey, with a few references, for example, Szeliski’s Computer Vision work [1].

# Approaches

Expanding on past approaches to steganography we work with multiple images instead of just one at time. This leads to a complexity in decoding, making the message a much harder task for those who do not have access to the decoder. This approach also uses the inherently variation of URLs generated by file sharing services to further add another layer of encryption to the data.

Our approach also addresses the other problem faced by image steganography (or encryption in general) wherein the key needed to decode the message needs to be either pre-accepted or sent separately increasing the likelihood of the message being intercepted and successfully decoded. We solve this by sending both the key to decode the message and the encoded message at the same time. We also use the traditional method of pre-accepted keys to further add another layer of encryption.





Where:

= Output RGB value

= RGB value of original image

= Data encoded in RGB format

In all our approach has three layers of encryption securely storing the information.

# Implementation

The Implementation is divided into 3 parts, namely

1. PreProcessing
2. Encoding
3. Decoding

The PreProcessing stage provides data for the both the Encoding and Decoding stages. The Encoding stage encodes the data into the image set and the Decoding stage does the inverse process used in Encoding.

## PreProcessing

The first step in our process is the images, for the purpose of this report we use lossless image storage formats like BMP or PNG. These are then stored in a folder and the program reads the images from this source folder and then generates an output folder with images which can be uploaded. These steps are explained in detail later on this this report.

The second step in the generation of a URL from a file sharing service, this report uses Google Drive, but any other file sharing service maybe used. Once the URL is generated it is decomposed into 4 segments as shown in below in Fig. 1

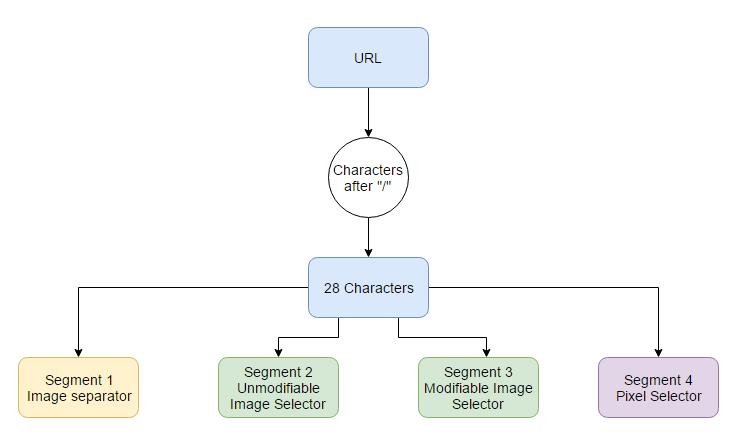


Figure 1: Preprocessing Block Diagram

### Segment 1

This Segment consist of the first 8 characters in the URL after the “/”. And are used to separate the image in the source folder into “Modifiable” and “Unmodifiable” categories.

Images in the Unmodifiable category are used to provide a benchmark (aka ) which is used in (1) to edit the RGB values in the Modifiable category.

### Segment 2

The next 7 characters in the URL are used to determine how an image is selected in from the Unmodifiable category. These images are selected using the following formula and an array is generated.

### Segment 3

The last 7 characters in the URL are used to determine how an image is selected in from the Modifiable category. These images are selected using the following formula and these are later added into and an array which the Encoder uses.

### Segment 4

The remaining 6 characters in the URL are used to determine how the pixel which is to be changed is selected. This leads a specific problem wherein the said pixel value may not exist in either one of the two images. This is tackled by ignoring that pixel and moving on to the next value determined by the URL.

## Encoding Process

The encoder takes the message to be encoded from the user and then uses the data from the arrays generated by the four segments in the preprocessing stage selecting images from the Unmodifiable and Modifiable sections and moving across pixels and images till all the data is encoded. Pixels which have been modified are marked with a flag bit which ensures previously written data is not lost by overwriting. Fig 2, shows the dependency of the Encode on other functions.

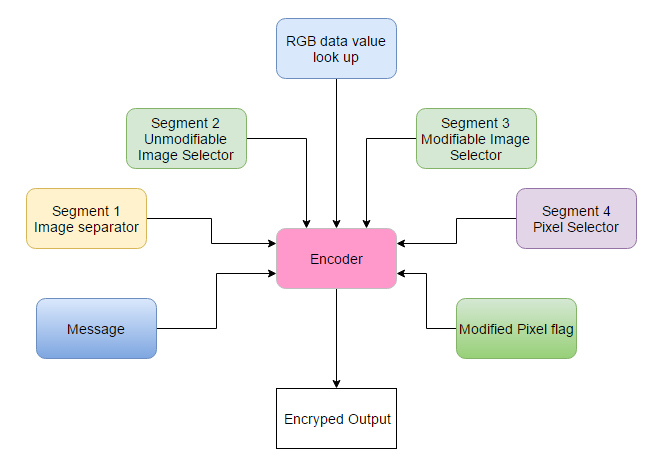


Figure 2: Encoder Dependencies

The data is encoded using a simple comparison table where ASCII characters are assigned a thee digit value, this value is then used as the in (1) This process can lead to 3 outlier conditions which are

1. Unknown Input character
2. End of message declaration
3. value is greater than 255

### Unknown Input Character

Modern Unicode text contain far more characters than traditional ASCII. Most messages can usually be successfully transmitted using ACSII characters but there may be situations where an unknown Unicode character is provided as input, this situation the program defaults all unknown characters.

### End of Message Declaration

As the program finishes encoding all the data, we need to be able to determine a stop signal to show the end of message. This is done by using the Unicode character “§” which does not typically occur in text. The program ends once it encounters the above symbol and generates a folder with all the images which can then be uploaded to in to the URL and shared.

### RGB Value out of bounds

Sometimes there maybe a situation wherein using (1) leads to a value greater than 255. But image formats default any value over 255 to white causing loss of data. This is avoided by checking if value is greater than a threshold determined by the larger value in .

## Decode

The decode process is very similar to the Encode process but it just uses a variation of (1) with the “+” sign changed to “-”. The decoding function use the data provided by the Preprocessing stage to decode the message hidden in the images.

# Results

The program is able to store messages of sufficiently large length. This length is only limited by the total number of pixels for example a standard HD image contain 737280 pixels. Meaning a string of the same length can be encoded into a single image.

## Encode

The Encoding program starts by requesting for the directory in where the images are stored (Fig 3), it then requests the URL (Fig 4) which is used to share the encrypted images. Finally the program asks the user for the message to be encoded into the image set (Fig 5). The program then exits with a dialog box showing the completion (Fig 6).

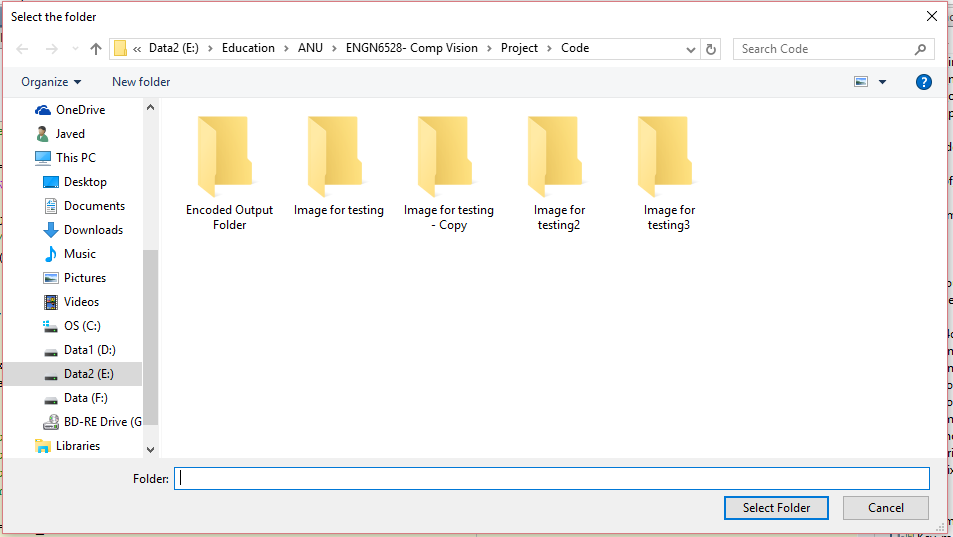


Figure 3: Directory Selection

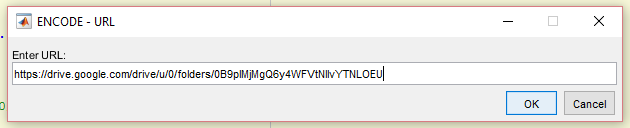


Figure 4: URL Input

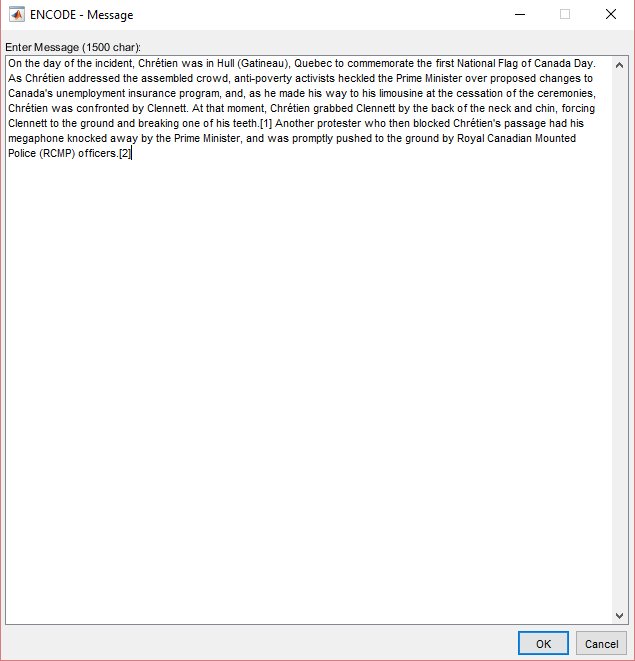


Figure 5: Message input Dialogue

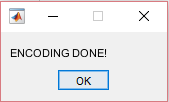


Figure 6: End of Encode Dialogue

## Decode

The Decode program starts by requesting the user to locate the directory where the images to be decoded are located (Fig 7). The URL which contains the key to decode the messages is then requested (Fig 8). The Decoded message is then displayed in a Dialogue box (Fig 9).

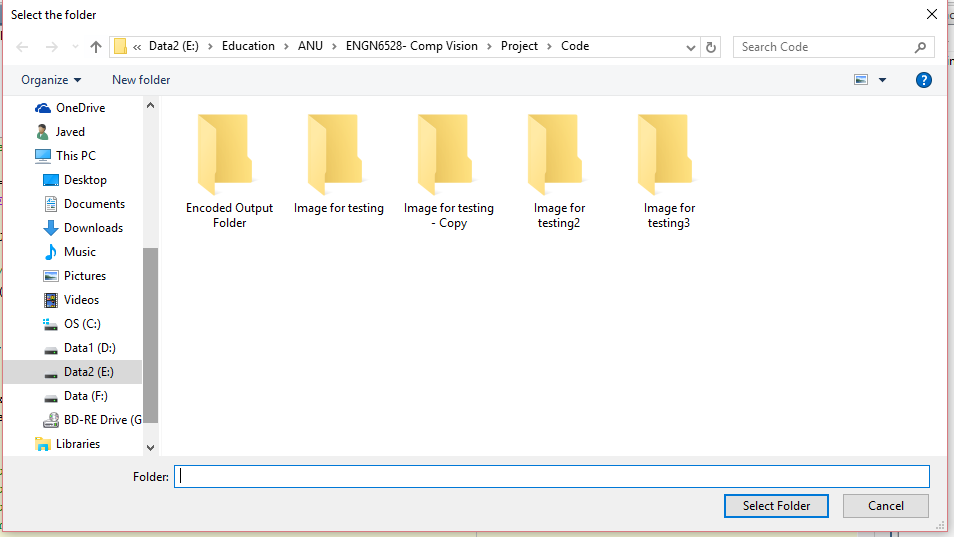


Figure 7: Decode Directory Selection

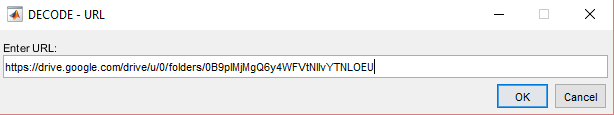


Figure 8: Decode URL input

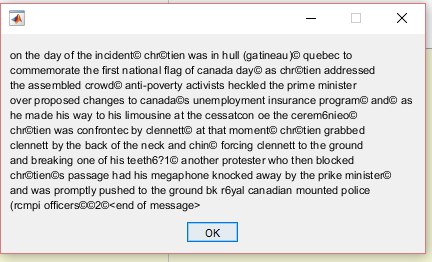


Figure 9: Decoded Message

## Image Data

As the image data is directly modified it is of particular interest to see how a modified image looks in comparison to its unmodified state (Fig 10,11). We performed a similarity analysis on the images based on a Message of length 72 characters. Fig 12 shows that they are similar in the orders of magnitudes.



Figure 10: Image before Encoding



Figure 11: Same image after encoding

Figure 12: Similarity plotted Vs Number of images

# Discussions

This method’s strongest suit is the use of a large number of images to hide a small number of characters of data. This also points at its weakness wherein if a very large message is to be encoded into a small image set it will lead to output images looking clearly garbled and its value as a steganography method will be undermined. This method is currently limited to the use of only lossless image formats and data lossless file sharing methods meaning the most common image format JEPG cannot be used.

Another limitation of this method is that while choosing the image pairs for encoding data, if the chosen pairs are widely different in can lead to noticeable points, especially on smaller images where this is more pronounced. The concept could be further developed to reduce the likelihood of such artefacts.

# Conclusions

In conclusion, this concept improves on existing steganography methods by using more than one image and incorporating multiple levels of encryption. It turns a simple method of sharing pictures into a secure data transfer technique.

This can find application in many fields especially in the defense, and high corporate sector.

# Learning Outcomes (non-assessable)

Please provide at least 3 learning outcomes and experiences about what you have learned from doing this project.

References

1. Szeliski R., *Computer Vision: Algorithms and Applications*, Springer, 2011.

Appendix (non-assessable)

Any appendixes, such as algorithms or pseudo codes, appear here, if necessary. It is not counted towards the page limit. Appendix is also useful for your future reference.

1. [↑](#footnote-ref-1)