# DATA ENCRYPTION USING LSB

Marcos Barbieri

200-53-364

Professor Rivas

MSCS-630L

Marcos Barbieri
Professor Rivas
MSCS-630L
Project Proposal

## Abstract

The proposed project will be designed to encrypt messages within images using concepts presented in the Least Significant Bit (LSB) and Advanced Encryption System (AES) algorithms. The app will have a similar design to a photo upload application. The focus of this project has shifted slightly since its proposal. As the project proceeded, further interest was found in encrypting messages within images. This is something that is not commonly used, and as a methodology is interesting to test. Therefore, the idea is to encode a message and place it within an image to provide obscurity.

## Introduction

The intent of this methodology, is to apply some of the concepts in image steganography. This means that the sender of a message will encrypt a message within an image. Images are often disregarded as means of transferring information. Thus, this makes them great for the encoding of private information. In addition, it can be a form of layered security. This means that after encrypting a message, said encrypted message can also be encoded into the image, creating more work for an attacker with intent of decrypting any messages being transmitted.

## Background

### Terminology

It is important to understand some terminology before proceeding with the argument. Each term defined has a definition specific to this project, but may or may not be the accepted use of the term in other projects. That being said

**Plaintext** refers to the original message that the user wrote down

**Cover Image** refers to the image that will contain the hidden data

### General Information

The native iOS programming language, Swift, uses an object called a `UIImage` to represent image files on an application. Through the `UIImage` object we can obtain pointers to the image in memory and obtain the bytes that make up said image. These bytes represent the RGBA codes that make up the pixel. The acronym RGBA represents the color space and stands for Red Green Blue Alpha, in which each corresponding code corresponds to each word in the acronym. Thus, there is a byte value for the amount of red, green and blue in the pixel as well as a byte for the alpha channel information. For this use case only the red, green, and blue channels will be used, but any implementation is not limited to using the alpha channel bit.

## Methodology

### Least Significant Bit (LSB)

This algorithm uses the Least Significant Bit (LSB) substitution technique to encode data sets. The process of LSB identifies a set of bits that can be regarded as non-essential to the quality of the picture identified by ***N***. Thus, we can consider ***M*** as our plaintext (unencrypted message) were $\sum_{i=0}^{n} \mathrm{N(i)} = \mathrm{M(i)}$ . This means that all of the "least significant" bits in our set ***N*** will be replaced by the corresponding amount of bits in ***M***.

### Channel Switching

An LSB-only approach can be easily deciphered once an attacker figures out that a message is encoded in an image. Therefore, to increase confusion, there must be a "channel switching" operation that needs to be made. However, this will require a key to be constructed. To follow the approach of AES, round keys can be created in the same way they are created in AES. The constructed key can identify the channel that the data is actually encoded on. Therefore, a simple modulus operation can calculate the channel. Consider the channel space $\boldsymbol{G}$, it is clear that the data must be encoded in one of the channels within the channel space, denoted by $\boldsymbol{G_i}$. $\boldsymbol{G_i}$ can be calculated by taking one byte of the constructed key, denoted by $\boldsymbol{K_i}$, and performing $\boldsymbol{K_i}$ % $\boldsymbol{n}$ where $\boldsymbol{n}$ is the number of elements in set $\boldsymbol{G}$. Thus, each pixel within the cover image will have embedded data at different locations in each byte.

## Experiments

As of now all experiments performed to test this algorithms have dealt with the retrieval of binary image data. Since iOS is strict about accessing memory, there are several stages that must be dealt with in order to buffer each pixel of data from the cover image. Currently, work is being done on actually editing the channel information on the cover image pixels.

## Analysis

After running tests on editing images with a static change (i.e. without channel switching), it is clear that there is no simple workaround to "refreshing" an image after its pixels have been altered. In other words, after encoded data has been written to the pixels, there is no evident data that is appearing on the image viewer widget on the user interface. The suspicion for this is that the pixels that were edited are actually a mutable copy of the pixels in memory and must therefore be re-displayed in the image viewer. This is most likely because iOS does not allow the unsafe writing to specific bytes of memory through simple objects like `UIImage`.

## Conclusion

Ultimately, it is clear that more work needs to be done to verify that the pixels are actually being written to in the cover image. Depending on the amount of bits that are changed in each channel, the distortion of the image may vary. Thus, further experiments must be run to examine the significant or insignificant distortion of the image after the message is embedded. In addition, the channel switching feature must be added after round key generation is created.

## References

Savani, Hussain. "Image Steganography." *LinkedIn SlideShare*, 2 Jan. 2014, www.slideshare.net/hussainsavani/image-steganography.

Pandey, Aishwarya, and Jharna Chopra. "Steganography Using AES and LSB Techniques." *International Journal of Scientific Research Engineering & Technology (IJSRET)*, vol. 6, no. 6, June 2017, pp. 620–623.