**Project Report**

**Names: Anthony Golubski-Allen,**

**Ibrahim Sardar,**

**Yousef Alsaadi**

**Project: Steganography**

**Date: 4/20/18**

* **Introduction**
* **Background**

We did our research into image steganography. We looked at different image steganography techniques that would allow us to embed images and text into another image. There were list of methods that we found which were classified into two domains. The spatial domain and the frequency domain. The spatial domain is where you directly change the values of the pixels. That is where we found the least significant bit algorithm. The frequency domain is where the images are changed by transforms and then the message is embedded. In that domain, we found out about the discrete Fourier transform and discrete cosine transform algorithms. We looked at each algorithm that we found to see which algorithm to use for our program within the semester. We would decide to go with the least significant algorithm because it was the most common one to use.

We would learn the weakness of the least significant bit algorithm does not take much to ruin the hidden object inside the cover image. The hidden image can be destroyed if you rotate, crop, draw, resize, and other ways that manipulate the stego image. This means that this algorithm will limit the stego image to be uncompressed or be in a format with lossless compression. Lossless compression keeps the image data intact while lossy compression will lose some data. We decided that we should use bitmap for our program for the mean time and add more formats later on.

The programming language that we decided to use for our project is C++. Everybody was familiar with C++. We decided to use free source library called WxWidgets for our graphic user interface. We used another free source library called CImg to be able to read and write to the bitmap image. We decided to go with CImg library because it was simple to add to our program. Also, CImg could use more image formats along the way once we installed the libraries to our computer. CImg has macros built in their code that will use those libraries if they are installed.

* **Implementation details**

**Graphic User Interface**

We would set up the graphic user interface first. There are four forms in our program. You can access these forms via file menus. We use the event table to set up the functionality of the buttons and file menus for each form. We set up a function that uses the least significant bit algorithm and based on what form that you are will use the algorithm discussed below.

**Text to image:**

We use LSB algorithm to implement this project in C++, in text to image, we wrote a function that read text and convert it to binary using bit set<8> in C++, then add each digit of the binary to a vector. Each cell has one digit of the binary number of the text, so now we have a vector with binary of the text or message that we want to encrypt.

After we convert text to binary and push it to the vector, we wrote another function to hide the text in the image, so we load the image it is in bmp format and loop through each pixel, as we loop through each pixel we have a nested for loop to go through the RGB of each pixel and get their values, then we convert their values to binary.

Now we have both, text and image in binary format, we read through the RGB binary and find the least significant bit, then we change it to the first bit in out text binary that we have stored in the vector. After we change the LSB we create new pixel and put the new value to it. Finally, we use draw point function to draw back the new pixel to the image and now we have the new encrypted image, to decrypt we just reverse the process but we read from image and store the message.

**Image to image:**

Same idea as in text to image about converting char to binary, but in image to image we used two least significant bits, so we loop through each pixel in the hide and covered image and find the 2LBS of the covered image, then

change the 2LSB of the covered image with the binary bits in the covered image, then we create new pixel that have the new values, then we used draw point function to draw back the new pixel to the covered image, for decryption we reverse the process

**File to image:**

* **Testing/Results**

**Text to Image**

**Encrypting**

**Before**

****

**After**

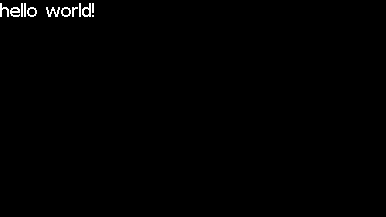
****

**Decrypting**

**Before**

****

**After**

****

**Image to Image**

**Encrypting**

**Before**

****

Figure :Image to Hide



Figure : Cover Image

**After**

****

**Decrypting Image**

**Before**

****

**After**

****

**File to Image**

**Encrypting**

**Before**

**After**

**Decrypting**

**Before**

**After**

**Video is in this folder directory**

* **Future Work**

Future work that we could do with our program is to add different image formats. We would have to use the discrete Fourier transform or discrete cosine transform to make sure that the jpeg format would work to hide an image. If we did not do that, the information would be lost if we used it with least significant bit algorithm.

Another option that we can do for future work is to add different algorithms for the user to choose to use to hide an image. We found many different algorithms that you could hide data in the third reference link. This would give the user multiple options to pick an algorithm that they would like to use.

* **References**

1. <http://www.sciencepub.net/researcher/research0212/12_2667research0212_67_73.pdf>

2. <https://zdoc.site/image-steganography-and-steganalysispdf-school-of-computing-.html>

3. <http://www.ijeit.com/Vol%203/Issue%207/IJEIT1412201401_22.pdf>