Class-Activity: Steganography

By Dr. Zahid Anwar

1 OVERVIEW

In this lab lesson, we'll practice with different tools to learn about the strengths and limitations of steganography for hiding the information. We will use the following two tools during the activity:

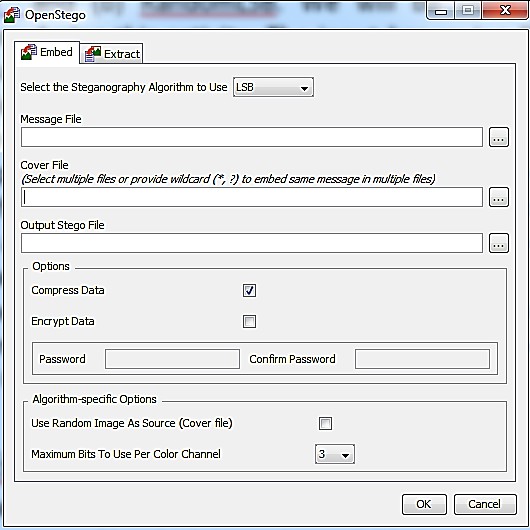
**Openstego:** A java executable file to hide and recover the information

**ImageComparator:** for deep inspection of apparently similar images.

2 OPENSTEGO OVERVIEW

Openstego is a java executable tool, which has been provided for the activity. The tool can be used to embed secret images inside a cover file and then recover the same.

Double clicking the jar file will execute the tool and display a graphic user interface as shown below:



The tool offers two main algorithms for embedding the secret (a) LSB and (b) RandomLSB. We will be using the LSB algorithm during this activity. The input bar saying “Message file” is used to browse through and select the secret image to be hidden, while the “Cover file” input can be used to select the cover file to be used for hiding the image. “Output Stego File” input requires the complete path and the name of the output file to be filled. You can also use the adjacent button to browse and select the desired folder and enter

the name of your choice for the output file. The “Option” pane can be used to compress or/and encrypt the data using user-supplied password. One of the important features offered by the tool the ability to define the number of maximum bits to be used for hiding information per color channel of the cover file. This flexibility is offered under the “Algorithm-specific Options” pane.

2.1 EMBEDING SECRETS USING OPENSTEGO

Let us attempt to embed different images inside the same cover file.

1. Use **“LSB”** as the steganography algorithm to encode the picture. .



2. Browse to the working directory and select “secret1.jpeg” as the message file.

3. Select “lena.png” as the cover file

4. Write “output1.png” as the name of the resulting output file.

5. Keep the option of “compress data” selected and

6. select maximum bits as 1.

7. Press OK to embed the secret (secret1.jpeg) inside the cover files (lena.png).

Q-1: Was the embedding successful? Open the output file (output1.png) and compare with the original lena picture. Increase zoom level +500 Are you visually able to discern any difference between the two files?

**Yes, the embedding was successful. No, I cannot visually see any differences between the original lena file and the output1 file.**

Now Lets try another one:

1. Fire up openStego again.

2. Remember to use “LSB” as the steganography algorithm to encode the picture.



3. Now select the “secret2.png” as the message file and repeat the same process (using 1 bit per color channel maximum), saving the output as “output2.png”.

Q-2: Were you able to embed the 2nd image? How or Why not?

to select the It offers two main algorithms various options for is file Message digests as discussed

**No, I was not able to embed the second message. This is because the image size is not enough to embed the data**

Q-3: Now increase the number of maximum bits to use per color channel to embed the same image.

Name the output file as “output2.png”. Use **“LSB”** as the stego algorithm.

. What are the **minimum** number of bits required per channel to embed the message file named “secret2.jpeg”? Fill in the boxes below after looking at the file properties



* The minimum number of bits required per channel to embed the message file named “secret2.jpeg” is 4.

1. Total Pixels of are W x H= WH

2. Each pixel has 4 channels (PNG) so 4 bits of hiding space

3. Total Storage = (4 x WH) / 8 bytes = 0.5 WH bytes

4. Total Storage of “lena.png”:

512 x 512 x 0.5 bit ≈ 131,072 bits = 131,072 /1024 =\_ 128 kbytes

5. The size of “secret2.png” is: 128 kb

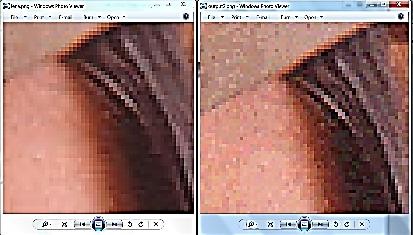
6. When 1 bit per color channel (byte) is used, the total number of bits of secret2.png is roughly

4 times the bytes available in lena.png and therefore the cover file failed to hide the image.

7. How many bits per channel are needed? 4

8. Try encoding with openStego with the number of bits selected calculated in 7. Does the encoding work? YES/ NO

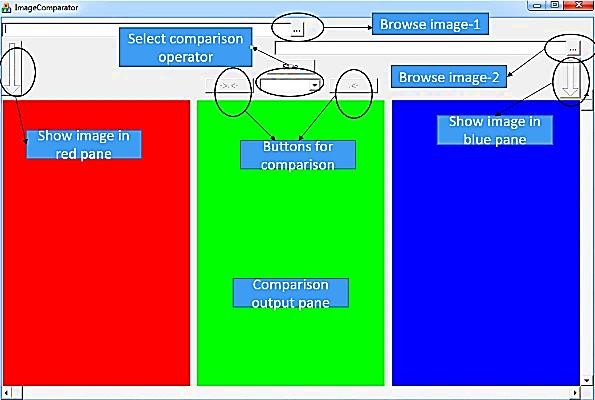
Q4: Visually compare the output files developed after hiding 1st (output1.png) and 2nd (output2.png resulting from question-3) images. Can you visually mark some differences between the quality of the two images with reference to the cover file “lena.png”?



**output2.png has a greater variation in pixel color when zoomed into the exact same spot, but is barely noticeable**

3 COMPARING USING IMAGE COMPARATOR

Image comparator is a tool (provided in the folder) to compare the difference between the two images. The tool the two images pixel by pixel and mark the differences between them through application of simple Boolean operations. We will be using this tool to investigate the difference between our reference lena image and the output files. The GUI of the tool along with hints on how to use its various features are shown in the figure below.



We will select the X-OR operator for comparing the two

figures. The truth-table of the X-OR is given here. The table says that whenever the input bits are

different the output of X-OR will be 1 and zero other-wise. This feature can be used

to compare the two images, pixel-by-pixel and isolate those pixels which differ between the two. Select and load the reference lena image in the red pane. Now load one by one the output files in the blue pane (first output1 and then output2) and perform the “X-OR enhanced” comparison by pressing the button. The result of the comparison will be displayed in the green pane.



A B X-OR

0 0 0

0 1 1

1 0 1

1 1 0

Q-5: (a) Has the tool identified some difference between the two files?(Yes/No) (b) What does the black portion represent in the output?

(c) What is the distribution of dissimilar pixels between the reference and color files (uniform all over the picture or focused on one area)?

(d) If focused, which part of each figure has been used to hide the data (upper or lower portion)?

(e) Also write the rough percentage of the cover file for hiding (a) secret -1 (b) secret-2. Why do you think these percentages are not the same for the two files?

**a.) Yes**

**b.) the flipping of bits to hide the image**

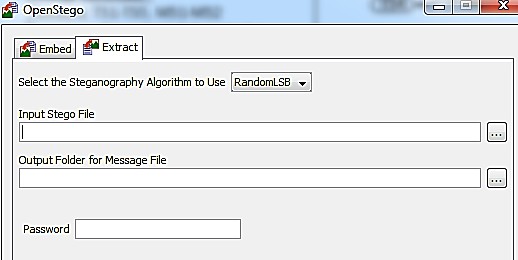
**c.) uniform all over the picture**

**d.) upper portion**

**e.) The fuzz percentage is about 3%. The percentages are not the same for the two files because two images need two different percents to be able to detect a difference.**

4 RECOVERING THE HIDDEN FILE USING OPENSTEGO

Now let’s use the openstego tool to recover the original files from “output1.png” and “output2.png”. Go to the Extract tab and select “RandomLSB” algorithm as shown in the snapshot below:



Browse and select the “output1.png” as input stego file. Choose the working directory as the Output folder and then press “Ok”.

Q-6: Were you able to recover the hidden file from the output1.png? Why or why not?

**No, I got an error message that says “embedded data is corrupt OR invalid password has been provided OR no algorithm found which can handle the given stego file”**

Q-7: What option(s) is (are) required to be changed to successfully recover both the hidden images? What do you think is the difference between LSB and random LSB?

**Changing the algorithm to LSB rather than random LSB will allow a user to successfully recover both hidden images. I think the differerence between LSB and random LSB is that LSB changes the rightmost bit resulting in “+” or “-“ 1 in the value of the bit string which is the the smallest possible change. Random LSB would then be changing a random bit in the sequence(not necessarily the rightmost bit).**

**For example, the pixels that are distributed with random LSB it is completely random. While regular LSB the pixels follow the original look of the image where you can still make out an image.**