A Comparison of Image Steganography Techniques

Neivin Mathew, Robyn Rintjema, Steven Kalapos

April 10, 2017

1 Introduction

While both cryptography and steganography provide a way to securely send messages between two parties, the difference lies in how those messages are obscured. In cryptography the plain text message is secured in cipher text, and while this may be difficult to crack it is easily identifiable as a secret message. The goal in steganography is to hide the message within other data, so that if an attacker were to stumble upon it, they would not be aware that a hidden message exists.

Through analysis of Least Significant Bit (LSB), and Discrete Cosine Transform (DCT) steganographic techniques, we hope to determine if there is a tangible difference in their performance. The objective of this report and demonstration will be to compare these two image steganography techniques. The two implementations will be compared for effectiveness, message capacity and the impact of hiding a message on the cover image.

2 Steganographic Algorithms

2.1 Least Significant Bit (LSB)

LSB replacement is a simple steganography technique that involves swapping the least significant bit of each pixel's colour components with the bits of the message. Consequently, an image with more colour channels (RGBA, CMYK) can hide more data than those with fewer channels (RGB). The size of the message that can be hidden in an image is:

$$capacity_{LSB} = width \times height \times |colourChannels| \quad bits$$

Although the definition of LSB is confined to replacing only one bit, it can be expanded to replace two or more bits as well. Consequently, this will result in lossy modification of the pixel's colour values and degrade the image.

However, using the standard LSB algorithm will not degrade the image at all since applying it to a cover image will not modify the actual colour values as it only replaces the parity bit of each component.

LSB is easy to detect, and is the most well known steganographic algorithm. It is also susceptible to statistical attacks making it infeasible to apply in practice. Nevertheless, it is still an effective way of concealing data.

2.1.1 LSB Encoding

1. Load a copy of cover image and secret message
2. Validate that the message is small enough to fit within the cover image

3. Convert the secret message into its binary representation and prepend the length of the actual message so we know when to stop when decoding the image
4. Iterate through every pixel in the image:
   (a) Split each pixel into its red, green and blue components
   (b) Replace the Least Significant Bit of each component with a secret message bit
   (c) Stop if there are no bits left to hide
5. Write out the new image containing the secret message

2.1.2 LSB Decoding

1. Load the secret image and create an empty character array
2. Create a temporary buffer that is the size of a character (depends on character encoding e.g. 8 bits)
3. Iterate through every pixel in the image:
   (a) Split each pixel into its red, green and blue components
   (b) Read the Least Significant Bit of each component and write it to the temporary buffer
   (c) If the buffer is full, convert it to a character, write it to the message array and clear the buffer.
   (d) If the character read was the special character : that was used to delimit the message length, convert the previously read characters into the length of the message to be read
   (e) If the number of characters read matches the actual length, terminate
4. Write out the extracted secret message

2.2 Discrete Cosine Transform

Discrete cosine transformation depends on the DCT function:

$$X_k = \sum_{n=0}^{N-1} x_n cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)k\right] \qquad k = 0, ..., N-1$$

Some advantages of DCT is its robustness when it comes to image manipulation. It boasts the ability to use less bandwidth to send the image with the secret message as it can be compressed once created. However, DCT can be tricky to implement as a high degree of precision in the pixel values is needed. DCT also causes noticeable degradation in visual quality which may alert attackers to the presence of a secret message.

Additionally, because of the need to use 8x8 blocks of pixels to store message data, the DCT message size is severely limited. The size of the message that can be hidden in an image is:

$$capacity_{DCT} = \frac{width \times height}{8 \times 8} \quad bits$$

2.2.1 DCT Encoding

1. Get the cover image path and secret message
2. Load in the cover image
3. Convert secret message to binary representation and add the length of the message in characters to the beginning in the form: `length*message`
4. Split the image into red, blue and green colour channels
5. Working with only the blue channel. Break the image into 8x8 blocks of pixels
6. Subtract 128 from each block to normalize pixel values around 0
7. Run each block through DCT function
8. Divide each block by the quantization table given below
9. The DC coefficient is located in position [0][0] of each pixel block and is an average of the entire block. Retrieve the DC coefficient and replace the least significant bit with the corresponding message bit
10. Multiply the image blocks by the quantization table and add 128 to each block
11. Piece together 8x8 blocks of the new image
12. Merge the new blue channel with the existing red and green channels
13. Write out the new image containing the secret message

2.2.2 DCT Decoding

1. Get the secret image and load it in
2. Split the image into red, green and blue channels (message hidden in blue channel)
3. Break the blue channel into 8x8 blocks of pixels and subtract 128 from each block
4. Divide each block by the quantization table above
5. Read in the least significant bit of each DC coefficient in 8 bit blocks
6. Once * is encountered set message size to integer that came before separator

7. Read in 8 bit blocks from image, converting them to plaintext characters, until the length of the message matches the message size recovered
8. Return the decoded message

3 Testing Methods

To compare LSB and DCT, a standard PNG-24 image `lenna.png` sized 240x240 was used. All experiments were run on a standard 2014 Macbook Pro. The two techniques were compared in the following categories:

1. **Time Taken** – The time taken to encode the same `10kB` plaintext message into our image was recorded and compared.
2. **Message Size** – The maximum limit for size of the message that could be hidden within the standard image was analyzed.
3. **Visual Quality** – A qualitative analysis of the visual quality of the image was also conducted.
4. **Image Error** – The following measures of image error were used to compare the two techniques.

   - **Mean Squared Error (MSE)**

     The MSE between the encoded image and the original image was calculated using the below equation. An MSE of zero means less image deviation from original, while an MSE of more than 1 indicates less similarity (grows as the differences in pixel colours increases.

     $$MSE = \frac{1}{MN}\sum_{n=1}^{M}\sum_{m=1}^{N}[\ h(n,m) - g(n,m)]^2 \qquad for\ images\ g(n,m)\ and\ h(n,m)$$

   - **Structural Similarity Index (SSIM)**

     The structural similarity index(SSIM) is used to measure similarity between two images based on their luminance, contrast and structure. A SSIM value of 1 means there is no perceived difference in the images. The measure between two windows x and y of common size $N \times N$ is:

     $$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_x y + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

     where μ_x is the average of x

     μ_y is the average of y

σ_x^2 is the variance of x

σ_y^2 is the variance of y

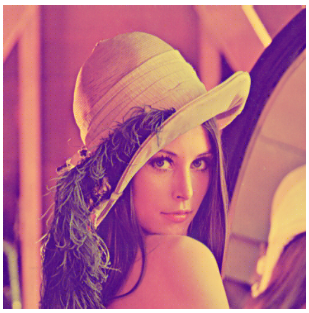
$\sigma_x y$ is the covariance of x and y

$c_1 = (k_1 L)^2, c_2 = (k_2 L)^2$ used as variables to stabilize division with weak denominator

L is the dynamic range of pixel values

$k_1 = 0.01$ and $k_2 = 0.03$ by default

4 Results

Result	Original	LSB	DCT
Image			
Time Taken	N/A	Encoding: 0.171s Decoding: 0.031s	Encoding: 0.234s Decoding: 0.075s
Message Size	N/A	21,600 bytes	450 bytes
Visual Quality	N/A	No quality difference	Slight blue tint Grid pattern present
MSE	0.0	0.02	1,886.86
SSIM	1.0	0.98	0.76

5 Conclusion

Both algorithms allowed for effective embedding and retrieval of the test message. Based on the results it can be concluded that the LSB algorithm provides a better image quality after embedding. This can be seen through analysis of the MSE and SSIM values as well as a visual inspection of the images. The time taken to encrypt and decrypt also favoured LSB, however the difference was negligible in a real world setting. However, LSB provides less security against attacks if the image is recognized as a steganographic image.

6 Tools Used

The project was implemented in Python and utilized the following libraries:

- OpenCV
- Pillow (Python Imaging Library)
- SciKit-Image
- NumPy
- Matplotlib

7 Bibliography

[1] M. Sravanthi, M. Devi, S. Riyazoddin, and M. Reddy, A Spatial Domain Image Steganography Technique Based on Plane Bit Substitution Method, Glob. J. Comput. , vol. 12, no. 15, 2012.

[2] S. Dhall, B. Bhushan, and S. Gupta, An in-depth analysis of various steganography techniques, Int. J. Secur. its Appl., vol. 9, no. 8, pp. 6794, 2015.

[3] F. M. Shelke, A. A. Dongre, and P. D. Soni, Comparison of different techniques for Steganography in images, vol. 3, no. 2, pp. 171176, 2014.

[4] G. Kaur and A. Kochhar, A Steganography Implementation based on LSB & DCT, vol. 4, no. 1, pp. 3541, 2012.