Effect of Noise on the Effiency of Steganography Methods

Hung Nguyen
The Bradley Department of Electrical and Computer Engineering, Virginia Tech
Blacksburg, Virginia, 24061
dnhung7@vt.edu

***Abstract*—This report describes the implementation of modeling additive white Gaussian noise on two steganography methods. Steganography is useful for cybersecurity and information concealment. Incremental analyses of changing variance was done on Least-Significant Bit and Spread Spectrum Image Steganography methods . The results indicate that that natural processes noise can have significant impacts on the decoded message contents through degradation.**

Keywords-component- steganography, LSB steganography, SSIS, spread-spectrum image steganography, Gaussian noise, white noise, signal corruption, encryption

I. INTRODUCTION

The implementations in this report attempts to model noise effects on the efficiency of steganography methods. Steganography is the process of concealing the existence of information within another form of information, especially images. This process is an improvement layer above conventional cryptography, where only the content of the information is concealed. The applications of steganography are useful across various industries such as digital copyrights and cybersecurity for communications [1]. Existing algorithms include, but not limited to, transform space and simple systems [3][4]. Diving into the real world information systems, noise corruptions from the effects of random processes in nature can have unfound effects like degradation on the communication of information, especially steganography [2]. Thus, it is desirable to model and analyze what effects the noises can have on steganography methods.

II. IMPLEMENTATION APPROACH

A. Steganography Methods

First, the information medium to be hidden is simply a message, implemented as a string in the MATLAB codes referenced in the Appendix [1]. This offers flexibility and ease of use in conversions to ASCII character codes for differential analyses.

Next, two steganography methods were selected for their simplicity: the Least-Significant Bit (LSB) Steganography method and the Spread Spectrum Image Steganography (SSIS) method [3][4].

For the LSB method, as the name specifies, utilizes the least-significant bit of each pixel in the image to conceal information [3]. This was implemented using the first pixel of the host image (the image that will be encoded with hidden information) as the message length, and the rest of the pixels as the contents of the image [6][7].

In the transform space, a simple SSIS by Lenarczyk was used and modified to encode and decode a string message instead of a simple generated watermarked image into the host image based upon a modulation and demodulation process in the transform space [5][11][12]. Note that both processes do not utilize encryption due to keep the effect of noise degradation simple.

B. Additive White Gaussian Noise and Encoded Images

In order to model random processes interferences in nature, a simple additive white Gaussian noise was implemented using MATLAB's internal *imnoise* function, with zero-mean and a specified variance. Thus, noise is added to the encoded image to simulate simple real-world processes' "degradations" before decoding the message [2][8][13].

$$\%\Delta Encoded\ Image\ Corruption = \frac{|(Encoded\ Image + Noise) - Encoded\ Image|}{Encoded\ Image} \quad (1)$$

$$\%\Delta Message\ Corruption = \frac{|Decoded\ Message_{corrupted} - Original\ Message|}{Orginal\ Message} \quad (2)$$

C. Analysis Methods

To analyze the effect of Gaussian noise, the variance of the added noise is varied over a relevant, adjustable range that seems to significantly change the corruption in the encoded image and the resulting corrupted decoded message, measured incrementally as shown in Equations (1) and (2) [9][14]. Implementing the incremental change in encoded image corruption required average the percentage change of every pixel. For the message corruption percentage change, the ASCII code conversions of the strings allow for useful numeric calculations [8][13].

III. EXPERIMENTAL RESULTS

The images, graphs, and texts produced from the implementation codes are shown below:

Figure 1. Original Image and Message

Figure 2.. Uncorrupted Encoded Images and Decoded Messages

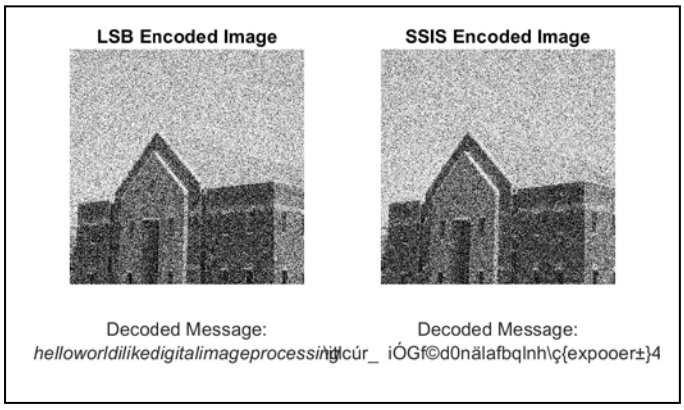

Figure 3. Example Corrupted Images (Variance = 0.1)

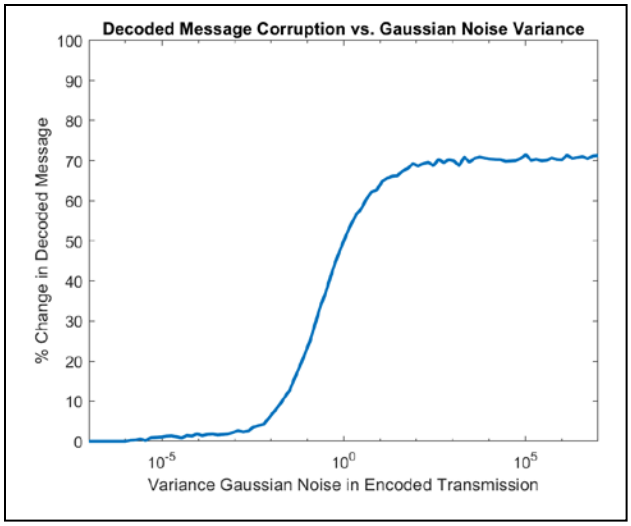

Figure 4. LSB Decoded Message Corruption vs. Gaussian Noise Variance

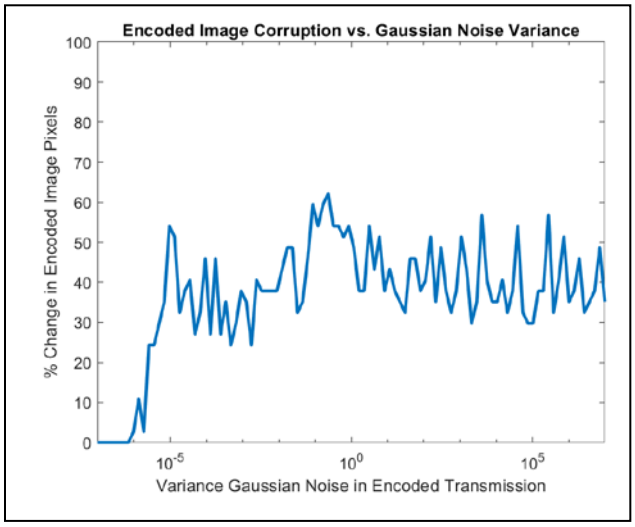

Figure 5. LSB Encoded Image Corruption vs. Gaussian Noise Variance

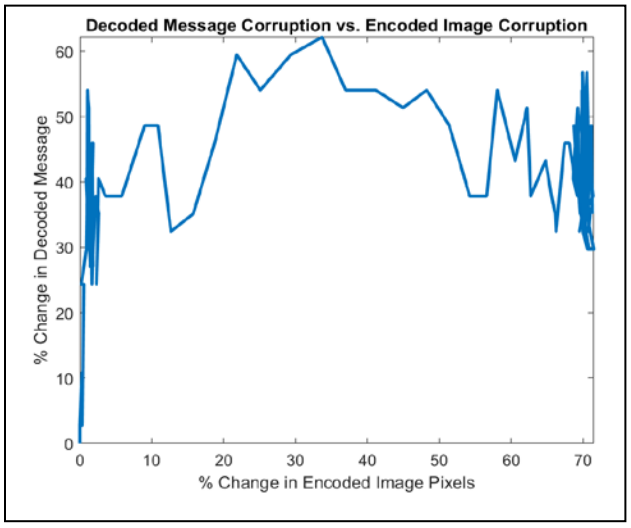

Figure 6. LSB Decoded Message Corruption vs Encoded Image Corruption

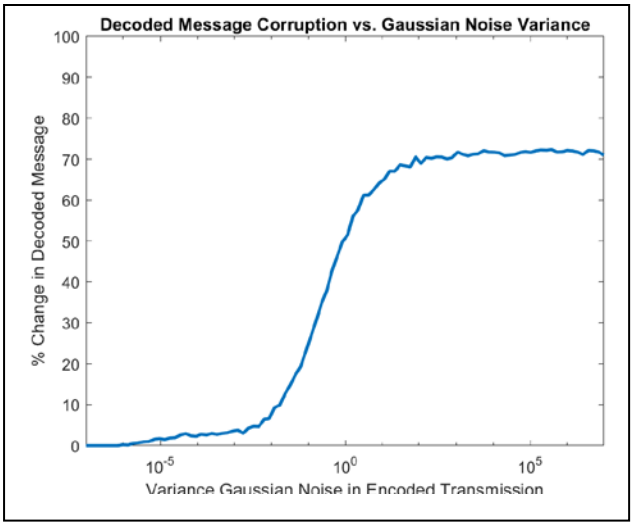

Figure 7. SSIS Decoded Message Corruption vs. Gaussian Noise Variance

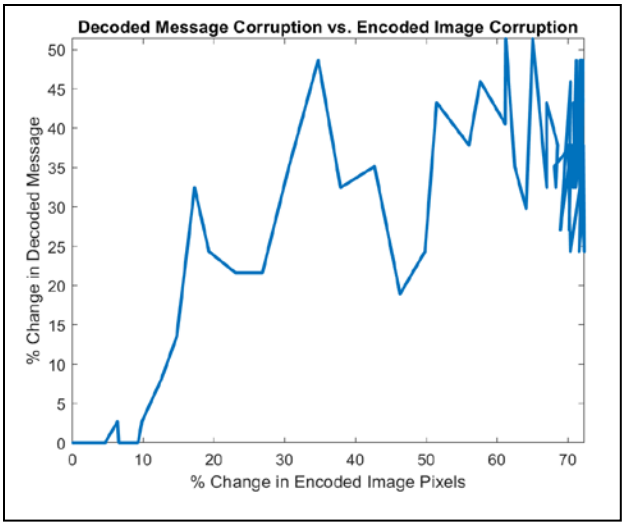

Figure 9. Decoded Message Corruption vs Encoded Image Corruption

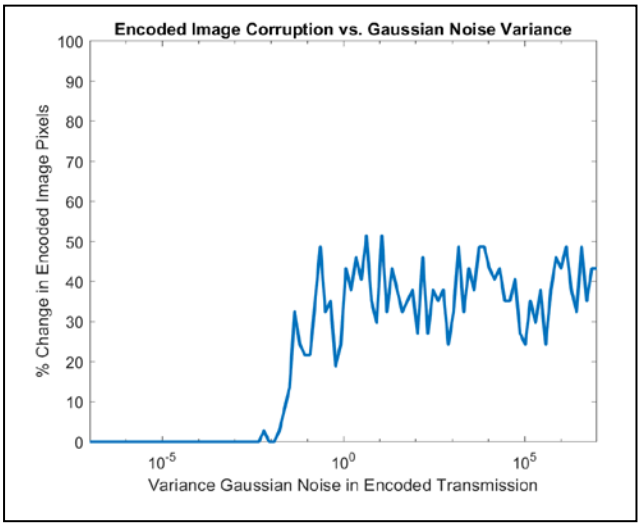

Figure 8. Encoded Image Corruption vs. Gaussian Noise Variance

IV. DISCUSSIONS

The images from *Figures 2* shows that the encoding and decoding implementations for LSB and SSIS are correct. The images from *Figure 3* illustrates what happens with introduced white Gaussian noise degradation. The scales for variance was modified to be logarithmic of 10 from $10^{-7} to\ 10^{7}$ due to the resulting sharp change in degradation of decoded messages near variances close to one (10^{0}) and flattening corruption changes closer to the opposite ends of the variance axis for both SSIS and LSB as shown in *Figure 4* and *Figure 7*. The encoded image degradations change seems delayed with respect to variance level for SSIS compared to LSB, which possibly means SSIS is somewhat resistant to noise as shown in *Figure 8* and *Figure 5*. Changes in message corruption with respect to encoded image corruption seems positively correlated in SSIS compared to LSB's flat high trend in *Figure 6 and Figure 9.* Overall, there seems to be a very sensitive degradation change in decoded message when variance white Gaussian noise is introduced.

ACKNOWLEDGMENT

Hung Nguyen thanks Dr. Jason Jianhua Xuan for this project and project partner Luke Segerdahl. Segerdahl completed most of the steganography implementations and revisions while also aided in the development of the efficiency analyses. Nguyen's personal contributions included refactoring implementations into functions and developing and implementing the data analyses and graphs.

REFERENCES

[1] R. J. Anderson & F.A. Petitcolas, "On the limits of steganography." IEEE Journal on selected areas in communications 16.4, 1998, pp. 474-481.

[2] R. C. Gonzalez & R. E. Woods, "Digital Image Processing", Addison-Wesley Publishing Company, Inc., 2018.

[3] M. A. B. Younes & A. Jantan, "A new steganography approach for images encryption exchange by using the least significant bit insertion." International Journal of Computer Science and Network Security 8, no. 6, 2008, pp. 247-257.

[4] L. M. Marvel, C. G. Boncelet, & C. T. Retter, "Spread spectrum image steganography," in IEEE Transactions on Image Processing, vol. 8, no. 8, pp. 1075-1083, Aug 1999.

[5] P. Lenarczyk, “Simple Spread Spectrum Watermarking Algorthm in Spatial Domain,” Sep. 24, 2015. [Online]. Available: https://bit.ly/2FPmNaO. [Accessed: May. 04, 2018]

APPENDIX

[6] *lsb_encode.m* – code for LSB encoding function

[7] *lsb_decode.m* – code for LSB decoding function

[8] *lsb_noise.m* – code for LSB noise degradation and single analysis

[9] *lsb_graphs_noise_variance.m* – code for LSB variance looping and graphs

[10] *practical_results_example.m* – code for example images

[11] *ssis_encode.m* - code for SSIS encoding function

[12] *ssis_decode.m* - code for SSIS decoding function

[13] *ssis_noise.m* - code for SSIS noise degradation and single analysis

[14] *ssis_graphs_noise_variance.m* - code for SSIS variance looping and graphs