A Comparison Study Using StegExpose for Steganalysis

Eric Olson, Larry Carter, and Qingzhong Liu

***Abstract*—Steganography is the art of hiding secret message in innocent digital data files. Steganalysis aims to expose the existence of steganograms. While internet applications and social media has grown tremendously in recent years, the use of social media is increasingly being used by cybercriminals as well as terrorists as a means of command and control communication including taking advantage of steganography for covert communication. In this paper, we investigate open source steganography/steganalysis software and test StegExpose for steganalysis. Our experimental results show that the capability of stegExpose is very limited.**

***Index Terms*—Steganography, steganalysis, stegexpose, openstego, virtual steganographic library.**

I. Introduction

Steganography is the practice of concealing a file, message, image, or video within another file, message, image, or video. The word steganography combines the Greek words steganos (στεγανός), meaning "covered, concealed, or protected", and graphein (γράφε ι ν) meaning "writing". The first recorded use of the term was in 1499 by Johannes Trithemius in his Steganographia, a treatise on cryptography and steganography, disguised as a book on magic. The advantage of steganography over cryptography is that the intended secret message does not attract attention to itself as an object of scrutiny. Plainly visible encrypted messages, no matter how unbreakable, arouse interest and may in themselves be incriminating in countries where encryption is illegal. Therefore, whereas cryptography is the practice of protecting the contents of a message alone, steganography is concerned with concealing the fact that a secret message is being sent, as well as concealing the contents of the message. Steganography includes the concealment of information within computer files. In digital steganography, electronic communications may include steganographic coding inside of a transport layer, such as a document file, image file, program or protocol. Media files are ideal for steganographic transmission because of their large size.

Our focus will be on freely available applications including VSL: Virtual Steganographic Laboratory [1] and StegExpose [2]. VSL is free image steganography and steganalysis software in the form of a graphical block diagramming tool. It can be used for complex testing and adjusting different steganographic techniques and provides a simple GUI and a modular, plug-in architecture. StegExpose is a steganalysis tool specialized in detecting LSB (least significant bit) steganography in lossless images such as PNG and BMP. It has a command line interface and is designed to analyze images in bulk while providing reporting capabilities. We have found little research conducted on VSL and StegExpose. These two open source tools will be used as our test applications.

We also used another graphical based steganography program OpenStego [3] to create steganography libraries for our tests since it can handle inputs of entire folders of images. This application is written in Java and has been tested on Windows and Linux. Openstego only allows for output to a PNG formatted steganography file. There are many other steganography software programs available and some we thought of including were:

Hide & Reveal [4] is an open-source steganography software and a java library distributed under the GNU GPL. It is written in Java and is multithreaded. It can perform encoding and decoding of BMP, PNG and TIF images.

StegSecret [5] is a java-based multiplatform steganalysis tool that allows the detection of hidden information by using the most known steganographic methods. It claims to detect EOF, LSB, DCTs and other techniques. It is distributed under the (GNU/GPL) license. Ben-4D Steganalysis Software [6] is written in Java and claims to apply a generalisation of the basic principles of Benford’s Law distribution is applied on the suspicious file in order to decide whether the file is a stego-carrier.

Steghide [7] is a widely used steganography software and has been considered standard in many of the papers we have researched. However, it has not been updated since 2003 according to the website. Qtech Hide & View [8] is an experimental application created by the KIT Steganography Research Group. This application is based on Bit-Plane Complexity Segmentation Steganography a new steganographic technique invented by Eiji Kawaguchi and Richard O. Eason in 1997. BPCS-Steganography is patented in USA (No. 6,473,516). This application is not open source, however it is available for limited use. OpenPuff [9] is an amazingly flexible program but included too many settings to choose from and did not easily create steganography images in batch production. Below we briefly discuss image steganography.

JPEG image steganography [10] techniques utilize the structure of this file type to develop various exploits. The JPEG is the most common image formant used on photo-sharing sites as well as on social media sites such as Twitter and Facebook and Google+. The encoding process consist of steps: applying lossy compression, division of the image into blocks, applying the Discrete Cosine Transform on each block and the quantization of the DCT coefficients. A greater

Manuscript received February 10, 2016; revised June 7, 2016. This work was supported in part by the Sam Houston State University (SHSU) Office of Research and Sponsored Programs, the SHSU College of Sciences and the SHSU Department of Computer Science.

The authors are with the SHSU Department of Computer Science, Huntsville, Texas 77341 USA (e-mail: ero003@shsu.edu, lec024@shsu.edu, liu@shsu.edu).

discussion of jpeg encoding and decoding can be found in [11].

Structure-based steganography exploits usually optional markers of the jpeg format. Examples include using the Exchangeable Image File or the Comments marker.

Spatial domain techniques typically modify the least significant bit (LSB) of the individual pixel values to embed the secrete data [12]. These methods exploit the fact that human perception is not sensitive to subtle changes in pixel values. This technique is not very reliable.

Frequency domain based methods-replace the least significant bits in the quantized discrete cosine transform coefficients. The DCT coefficients with a zero value are not used to avoid visual distortion of the cover image. [13]

There has been a multitude of research regarding steganalysis and steganography over the years and steganography has been used since ancient times. However, we did not find much research regarding software applications mentioned of VSL and StegExpose. Stegexpose was introduced in [2] and as well as a brief overview of the capabilities of the software for steganalysis. VSL was only mentioned in [14] as a tool to use for steganalysis and also cited in [15] but we were unable to access this paper.

Articles and papers [16]−[18] as well as other research has discussed how terrorists and cyber criminals have used steganography in videos, images, etc. to hide messages. Research in [19] explored the various social media platforms, analyzed prior research and found that some social media platforms supported steganographic images while others did not. According to [19] "At first glance, we see that most of the tools (except YASS) fail on Facebook and Flickr but succeed on Google+ and Twitter. Google+ is the most generous platform and accommodates all the steganography tools. Twitter is the next best; GhostHost fails on Twitter but the other tools are able to successfully exchange hidden content. Facebook and Flickr show the least compatibility with steganography in that all the tools except YASS (with high redundancy) fail in successfully exchanging secret content."

In this paper, we investigate open source steganography/steganalysis software and seek the possible application in the steganalysis of open source social media and online data. Our image library is described in section II experiments are presented in section III, followed by discussion in section IV and conclusions in section V.

II. Steganography Image Library

A library of 5150 raw color images in bmp format that have never been compressed [20]−[22] were used in our study. These images were converted to jpg and png files using XNview/Convert [23]. We created a small txt file of approximately 30 bytes named hidden_test.txt and inside this file stated "This is a hidden message......". We also used an approximately 14kb jpg screen capture (Capture.jpg) to use as another message file after using the txt file first. We than began the creation of our steganography image library by using Openstego on the original bmp images and batch processed the images. We chose OpenStego to use for the encoding process due to its ability to do batch processing. We encoded all of the raw images with the text file included with the image library download using the least significant bit technique (LSB) which is used by OpenStego. We also used VSL to create another image library of various formats to test additional detection capabilities. This library will then be used to test the steganalysis abilities of our chosen applications.

The original files were batch processed in OpenStego as described below:

- Original BMP files were processed as cover file and the small txt file was input as the message file with no password
- Original BMP files were processed as cover file and the small txt file was input as the message file with the password "test"
- Original BMP files were processed as cover file and the small jpg screen capture file was input as the message file with no password
- Original BMP files were processed as cover file and the small jpg screen capture file was input as the message file with password "test"

We chose random files within each folder and extracted the hidden information to verify that each file had embedded separately with the hidden message. OpenStego turns any files input into the cover and message files into .png formatted steganography output files.

We then used VSL to create another library of steganography images based on the encoding process used within VSL. VSL contains LSB, KLT and F5 algorithms for encoding files but the LSB option does not allow for a password to be input. After several attempts we were only able to make use of the LSB and F5 options within VSL as the KLT option continuously failed. We will only use the LSB option as the F5 option is not available in OpenStego.

VSL has the capability to input and output various file formats which makes it an excellent steganography image library builder. The original files were batch processed in VSL as described below:

- Original BMP files were processed as cover file and the small txt file was input as the message file with no password as no password is allowed. Steganography files were stored in jpg, bmp and png formats.
- Original BMP files were processed as cover file and the small jpg screen capture file was input as the message file with no password. Steganography files were stored in jpg, bmp and png formats.

III. Experiments on Steganography Image Library

A. Experiments with VSL

After several attempts to analyze files using VSL we quickly learned that it was unable to conduct steganalysis on files not created within VSL. The original files were all shown to have the following type of output from VSL "Estimated message size [B]:2338.3630853430477","no message". The OpenStego files were shown to have the following type of output from VSL "Estimated message size [B]:2374.347070592665","no message". This quickly became a problem when attempting to analyze any files that were not created within VSL. However, VSL was efficient at analyzing files created by it as shown by the following output "Estimated message size [B]:2320.691095311599","30". As seen in the last output from VSL it recognized the byte size of the small txt file (30 bytes) that was input as the hidden message. VSL also recognized the byte size of the Capture.jpg file (14kb) that was input as the hidden message, "Estimated message size [B]:5473.31172102455","14101". VSL was found to have a 100% detection rate of steganography files created in VSL but had 0% detection of steganography files created within OpenStego. We were also unable to use the BSM-SVM option within VSL as it continuously failed when attempting to analyze files and only the LSB-RS analysis option could be used.

B. *StegExpose Analysis of Untouched Files*

We used StegExpose for the final steganalysis test results due to the earlier limitations described when using VSL. StegExpose uses the command line and is run using the following format “java -jar StegExpose.jar testFolder default default steganalysisOfTestFolder” [2]. The default settings included are “speed” which is either at default or fast as and the other default option includes “threshold” which can be a floating point number between 0 and 1 analyzed the original raw images within StegExpose at default (0.2), .15, .25 and .50 threshold both in the .bmp format as well as in the converted .jpg and .png formats. According to [2], “The default value here is 0.2 (for both speed modes) and determines the level at which files are considered to be hiding data or not. A floating point value between 0 and 1 can be used here to update the threshold. If keeping false positives at bay is of priority, set the threshold slightly higher ~0.25. If reducing false negatives is more important, set the threshold slightly lower ~0.15.” Although this library of 5150 images have no any hidden data, a number of them were shown as suspicious and we see these as false positives. The results of the initial analysis on the original files without steganography are shown below in Fig. 1.

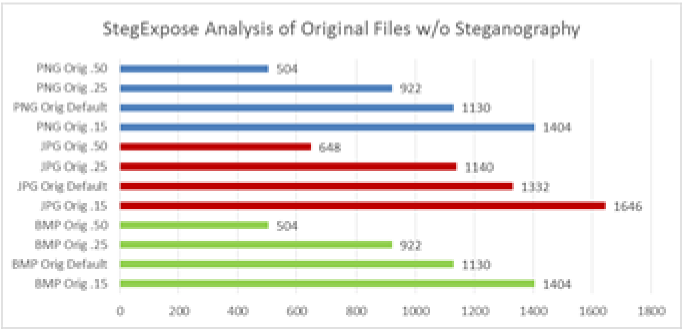

Fig. 1. Detection results by applying StegExpose to untouched files.

Each process above was run on the different sets of files including .jpg, .png and .bmp files without steganography under different thresholds. The lower the threshold in StegExpose the more files will be detected as suspicious. As we can see from the results the original .bmp files and converted .png files obtained the same results when analyzed in StegExpose while the converted .jpg files were detected at a higher rate. False positive percentages are shown in Fig. 2.

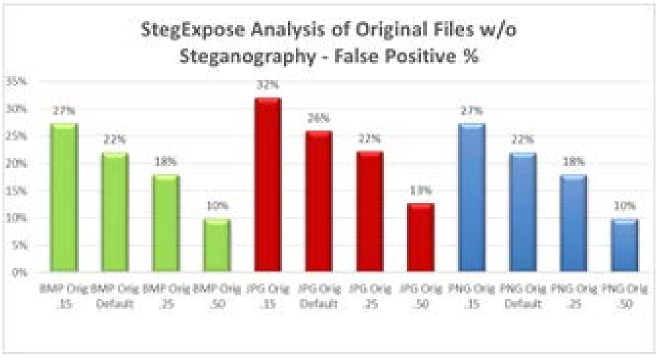

Fig. 2. False positive by applying StegExpose to untouched files.

C. *StegExpose Analysis of OpenStego Stego-Files*

We used StegExpose to analyze the library of steganography files created with OpenStego and the results for the embedded text file are shown below in Fig. 3.

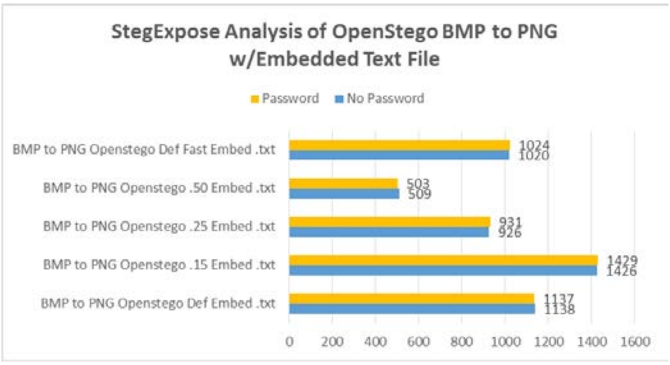

Fig. 3. Detection by applying StegExpose to stego-files (text hiding).

We can see in the chart above (Fig. 3) that there was little difference of detection by StegExpose when analyzing the OpenStego created files with a password or without. We also used the default threshold and ran one series of tests with the “fast” option in StegExpose and saw a reduction in detection from the default settings. Comparing the results of the text embedded files with the original files and we can see that there really isn’t much of a change in the detection rate.

Further analysis of the files created using OpenStego and embedding the .jpg image file into the files shows a much different result of detection than the embedded text file as shown in Fig. 4.

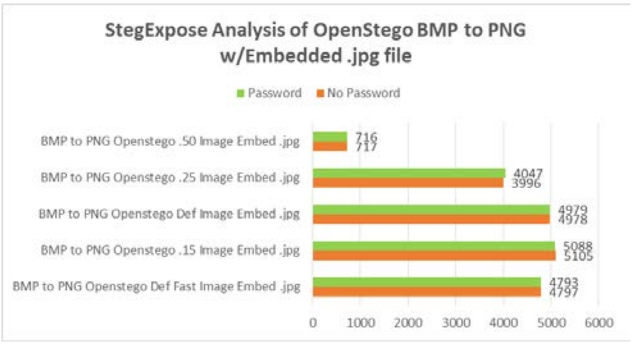

Fig. 4. Detection by applying StegExpose to stego-files (image hiding).

As we can see above in Fig. 5 there is a drastic increase in detection by StegExpose of the files created by OpenStego embedded with the approximately 14kb .jpg format image file. However, again the detection rate was relatively unchanged when including or not including a password with the embedded image. As shown in the above charts the Fast mode in StegExpose had a slightly lower detection rate than the slower default mode. However, according to [2] "A 460x460 pixel image will take 1.20 seconds to process in the default mode and 0.34 seconds in fast mode. However, the fast mode should be even faster in a real world environment, where there are a lot less stego files, allowing StegExpose to skip expensive detectors more frequently"

D. StegExpose Analysis of VSL Stego-Files

VSL was able to output to a variety of formats using the LSB technique. We output the steganography files to .jpg, .png and .bmp files after embedding the .txt or .jpg message files. The results for the .bmp and .png files appeared to match the StegExpose results for the OpenStego files but the .jpg output files were much different as seen below in Fig. 5.

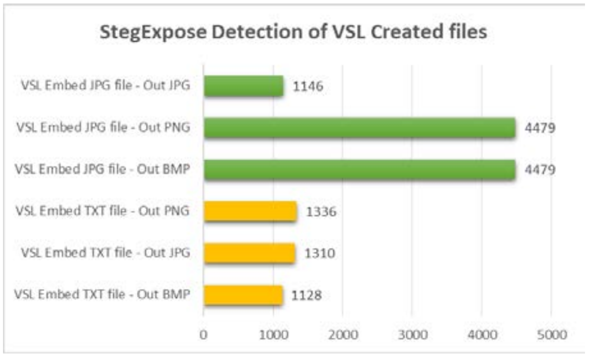

Fig. 5. StegExpose analysis of VSL stego-files.

The detection rates of StegExpose for the .png and .bmp files created in VSL have values close to the range detected by StegExpose on OpenStego created files. However, when embedding the .jpg file and outputting a .jpg file VSL appears to have an issue with the output. We ran this scenario a few different times to confirm the results and obtained the same results each time. We attempted to decode and analyze the files using VSL but it was unable to decode or analyze the .jpg files created by it after embedding the .jpg message file. VSL would output an error message when attempting to decode the .jpg files. This appears to be an issue with the VSL program again and not an issue with StegExpose. However, as stated before the .png files output by VSL appear to have similar detection rates as the OpenStego files output as .png files as shown below in Fig. 6.

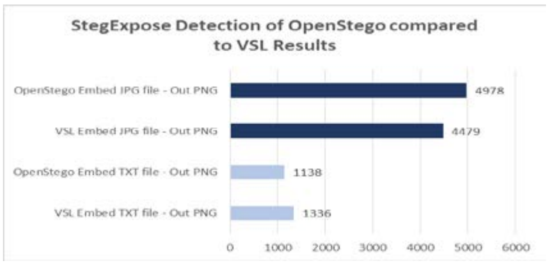

Fig. 6. Comparison of the detection on OpenStego and VSL.

IV. Discussions and Conclusions

StegExpose was used for steganalysis of the images using steganography we created as well as images downloaded from social media websites. We noticed, however that it produced a high percentage of false positives in its analysis of images that have not had steganography applied. The false positive rate ranged from 10-32% while the default settings were found to be 22-26%. The detection rates of StegExpose in regards to small embedded text files was minimal and not a significant change beyond the false positive percentages. However, the detection rates significantly increased when image files were embedded as the message file.

If we exclude the StegExpose threshold of .50 that we used for analysis we get a clearer picture of detection percentages for the threshold settings of .15, default and .25. StegExpose clearly excels at detecting LSB files embedded with images and the detection rate of images embedded with small text messages does not show any higher detection than a false positive as shown below in Table I (B).

TABLE I: StegExpose Detection Rate

(A) Thresholds. 15 - 50

Process (TH of .15 - .50)	StegExpose Detection %
Baseline – No Stego	10% – 32%
OpenStego – Embed txt file	10% - 28%
OpenStego – Embed jpg file	14% - 99%

(B) Thresholds. 15 - 25

Process (TH of .15 - .25)	StegExpose Detect %
Baseline – No Stego	18% – 32%
OpenStego – Embed txt file	18% - 28%
OpenStego – Embed jpg file	78% - 99%

We were disappointed with the results VSL: virtual steganography library. It has an easy to use GUI and we really liked the modular programming that offered ease of use and many options. However, its ability to analyze images we found to be poor at best. It was able to process large inputs and encode images in LSB or F5 with ease. We chose not to use the F5 outputs and analyze due to the results we were obtaining in the LSB outputs. OpenStego was also only able to output using LSB. StegExpose showed promising detection percentages when embedding images as the message file. However, it seemed to fail in detecting images embedded with small text files. The batch processing capabilities of all three pieces of software used shows that these tools can easily create large steganographic libraries. We used the clean image library from [22] as it was easy to obtain and download the images.

This leads to another issue we noticed when attempting to locate clean and steganography images is that there were various databases used by researchers. Many of these databases of images were unavailable or went to online links that were no longer working. A larger database of images could be used that have been processed with various steganography software to compare steganalysis results.

Much of the free or open source software available online tends to be outdated and several years old with little maintenance. However, StegExpose, OpenStego, and VSL have all been updated within the past five years. We believe further research is needed to evaluate the various open source and freely available steganalysis software available. These results could also be compared to steganalysis results from pay for software such as StegAlyzer [24] or StegoHunt [25]. We attempted to obtain a trial use of StegAlyzer as other researchers have used this in their paper. However, we were advised that the product is no longer supported or sold at this time. After our results were obtained we also attempted to analyze the images with a freely available program called StegSecret [5], but StegSecret had a zero percent detection rate with our library of steganography images. In other words, the capability of stegExpose is very limited in practical steganography detection.

REFERENCES

[1] M. Węgrzyn. (2015). VSL: Free steganography and steganalysis tool, Vsl.sourceforge.net. [Online]. Available: http://vsl.sourceforge.net/

[2] B. Boehm. (2015). StegExpose. GitHub. [Online]. Available: https://github.com/b3dk7/StegExpose

[3] S. Vaidya. (2015). OpenStego–Home. Openstego.com. [Online]. Available: http://www.openstego.com/

[4] N. Cottin. (2015). Hide & Reveal. Hidereveal.ncottin.net. [Online]. Available: http://hidereveal.ncottin.net/

[5] A. Munoz. (2015). StegSecret. A simple steganalysis tool. Stegsecret.sourceforge.net. [Online]. Available: http://stegsecret.sourceforge.net/

[6] (2013). Ben-4D Steganalysis Software. SourceForge. [Online]. Available: http://sourceforge.net/projects/ben4dstegdetect/

[7] S. Hetzl. (2015). Steghide.sourceforge.net. [Online]. Available: http://steghide.sourceforge.net/

[8] (2015). Datahide.org. Qtech Hide & View. [Online]. Available: http://datahide.org/BPCSe/QtechHV-program-e.html

[9] (2015). Embeddedsw.net. OpenPuff - Steganography & Watermarking. [Online]. Available: http://embeddedsw.net/OpenPuff_Steganography_Home.html

[10] S. Suri, H. Joshi, V. Mincoha, and A. Tyagi, "Comparative analysis of steganography for coloured images," *JCSE International Journal of Computer Sciences and Engineering,* vol. 2, no. 4, pp. 180-184, 2014.

[11] The JPEG committee home page. [Online]. Available: http://www.jpeg.org/.

[12] A. Cheddad, J. Condell, K. Curran, and P. M. Kevitt, "Digital image steganography: Survey and analysis of current methods," *Signal Processing,* vol. 90, no. 3, pp. 727-752, 2010.

[13] C. Chang, S. Chen, and Z. Chung, "A steganographic method based upon JPEG and quantization table modification," *Information Sciences,* vol. 141, no. 1-2, pp. 123-138, 2002.

[14] R. Das, "An investigation on information hiding tools for steganography," *International Journal of Information Security Science,* vol. 3, no. 3, pp. 200-208, 2014.

[15] H. Karaman and S. Sagiroglu, "An application based on steganography," presented at IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, 2012.

[16] S. Gallagher. (2012). Steganography: How al-Qaeda hid secret documents in a porn video. Ars Technica. [Online]. Available: http://arstechnica.com/business/2012/05/steganography-how-al-qaeda-hid-secret-documents-in-a-porn-video/

[17] (2015). United nations publication describes terrorist use of steganography-backbone security. [Online]. Available: https://www.backbonesecurity.com/TerroristUseofSteganography.aspx. [Accessed: 01- Dec- 2015]

[18] A. Bakier. (2015). The new issue of technical mujahid, a training manual for jihadis. The Jamestown Foundation. [Online]. Available: http://www.jamestown.org/programs/tm/single/?tx_ttnews[tt_news]=1057&tx_ttnews[backPid]=182&no_cache=1#.Vl0R_OLw--Y

[19] J. Ning *et al*. Secret message sharing using online social media. [Online]. Available: http://spirit.cs.ucdavis.edu/pubs/conf/ning-cns14.pdf

[20] Q. Liu and Z. Chen, "Improved approaches with calibrated neighboring joint density to steganalysis and seam-carved forgery detection in JPEG images," *ACM Transactions on Intelligent Systems and Technology,* vol. 5, no. 4, 2014.

[21] Q. Liu, A. Sung, and M. Qiao, "Neighboring joint density-based JPEG steganalysis," *ACM Transactions on Intelligent Systems and Technology,* vol. 2, no. 2, 2011.

[22] Q. Liu, A. Sung, Z. Chen, and J. Xu, "Feature mining and pattern classification for steganalysis of LSB matching steganography in grayscale images," *Pattern Recognition,* vol. 41, no.1, pp. 56-66, 2007.

[23] (2015). Xnview software for reading, organizing and processing images. [Online]. Available: http://www.xnview.com/en/

[24] (2015). SARC-Steganography Analysis and Research Center: Home. [Online]. Available: https://www.sarc-wv.com/

[25] (2015). Wetstonetech.com, WetStone Technologies, Inc–Shopping. Cart. [Online]. Available: https://www.wetstonetech.com/product/stegohunt/

Larry E. Carter served in the United States Navy (1969-1973) as an aviation electrician's mate. Assigned to the squadron VF-121 at NAS Miramar, San Diego, California USA. He worked on the McDonnell-Douglas Phantom II aircraft. He was honored in serving in this squadron that was the host of the navy's first fighter weapons school later becoming known as top gun. He has worked in many areas in the field of electricity from construction to the maintenance and support of highly automated processes in manufacturing and industry. He is currently employed as an instructor of electrical technology for Trinity Valley Community College, Athens, Texas USA. His current research interest is in steganography/steganalysis.

Eric Olson graduated from the Department of Computer Science of Sam Houston State University with a master's degree in digital forensics. His research interests include digital forensics, cyber security and single board computers such as the raspberry pi.

Qingzhong Liu is currently an associate professor at the Department of Computer Science of the Sam Houston State University. His research interests include multimedia forensics, information assurance, data mining, bioinformatics, and intelligent computing applications.