2007

Steganalysis in computer forensics

Ahmed Ibrahim
Edith Cowan University, aibrahi0@our.ecu.edu.au

DOI: 10.4225/75/57ad58327ff2e
Originally published in the Proceedings of the 5th Australian Digital Forensics Conference, Edith Cowan University, Perth Western Australia, December 3rd 2007.

Steganalysis in Computer Forensics

Ahmed Ibrahim
School of Computer and Information Science
Edith Cowan University
aibrahi0@student.ecu.edu.au

Abstract

Steganography deals with secrecy and convert communication and today the techniques for countering this in the context of computer forensics has somewhat fallen behind. This paper will discuss on how steganography is used for information hiding and its implications on computer forensics. While this paper is not about recovering hidden information, tools that are used for both steganography and steganalysis is evaluated and identifies the shortcomings that the forensic analysts would face. In doing so this paper urges on what the stakeholders in the field of computer forensics needs to do to keep ahead of criminals who are using such techniques to their advantage and obscure their criminal activities.

Keywords

Steganalysis, Steganography, Information Hiding, LSB, Stegdetect, Steghide, Outguess, Chi-Square, Digital Invisible Ink Toolkit

INTRODUCTION

One of the first widely used method for secure communication was steganography, also referred to as secret writing. A variety of techniques such as the application of invisible ink and masking the secret text inside an inconspicuous text existed during the early days (Pieprzyk, Hardjono & Seberry, 2003). It even dates back to ancient Greeks who practised the art of hiding messages by tattooing onto the shaved heads of messengers. Today, steganography has taken new meaning and is referred to the science of hiding messages in different electronic media such as graphic, audio, and video files (Schneier, 2000).

Besides the reasons for covert communication to maintain secrecy, steganography is also used to protect intellectual property rights using watermarking techniques that embed a digital fingerprint in the media (Silman, 2001). While each of these aforementioned purposes of steganography has its own applications, this paper will be concerned with the former.

Historically, much attention has been given on cryptography to ensure the secrecy of confidential information whether it is for storage or communication, however in recent times, different motivations have led people to pursue even more guaranteed approaches. Krenn (2004) reports that terrorist organisations use steganography to send secret messages using websites and newsgroups according to claims by the United States government. Although there is no substantial evidence supporting these claims, one would wonder why such an approach maybe attractive and realistic for such organisations. Other sources such as Kelley (2001a, 2001b), and McCullagh (2001) have also made similar claims. According to Schneier (2000), the privacy offered by steganography is far beyond that provided by encryption. This is because the goal of steganography is to hide the secret while encryption simply makes the secret unreadable.

New steganographic techniques are being developed and information hiding is becoming more advanced based on the motives of its use (Krenn, 2004). Besides the hype of terrorists using steganography, very recently there has been a case of corporate espionage reported by Phadnis (2007), where confidential information was leaked to a rival firm using steganographic tools that hid the information in music and picture files. Although the perpetrator was caught in this case, it does give an idea of the wide landscape in which steganography can be applied in.

This paper will focus on steganography of graphic or image files. It will describe some technical aspects of steganography used by different tools that are specific to certain types of image files. Following that, steganalysis techniques used to detect the presence of hidden information from the forensic analyst's point of view will be discussed. Finally, the limitations in steganalysis will be presented along with the evaluation of some steganalysis tools.

TECHNICAL PERSPECTIVE OF STEGANOGRAPHY

Steganography may be implemented using a variety of techniques and methods and a steganographic tool may employ any such method or a combination or even variations of such methods. These methods may range from the use of Least Significant Bit (LSB), manipulation of image and compression algorithms, and modifications of image properties such as its luminance (Johnson & Jajodia, 1998).

The use of LSB is the most commonly used technique for image steganography. Such tools are also referred to as image domain tools that manipulate the LSB using bitwise methods (Krenn, 2004). Since this is more like using noise to hide information in the LSB, small variations in the LSB are unnoticeable to the human eye (Wayner, 2002). However, one major limitation in the use of LSB is the amount of usable space to hide the secret message, thus a suitable cover image is essential. If the cover image does not satisfy the capacity requirements to hide the data, the steganographic image would appear to be suspicious (Krenn, 2004).

Furthermore, the success is also dependent on a reliable compression algorithm to ensure that the hidden message is not lost after the transformation (Krenn, 2004). According to Silman (2001), the most commonly used compression algorithms are Windows Bitmap (BMP), Graphic Interchange Format (GIF), and Joint Photographic Experts Group (JPEG). When LSB method is used, lossless compression algorithms such as BMP and GIF are preferable. This is because lossy compression algorithms such as JPEG are mainly used to save on storage space due to the fact that the compression gets rid of unwanted noise, limiting the amount of space that can be used for steganography (Wayner, 2002).

For lossy algorithms like JPEG, usually when the image is 24bits or grayscale, a more robust approach is to use masking/filtering where the luminance of parts of the image are modified. A more complex way to hide information, particularly in JPEG files is to use Discrete Cosine Transformations (DCT). DCT is also used by the JPEG compression algorithm, and the resulting steganographic image does not have any detectable visible changes as the technique makes use of the frequency domain of the image (Krenn, 2004).

STEGANALYSIS TECHNIQUES

Steganalysis is mostly about the discovery (Silman, 2001) and simply identifying the existence of a hidden message (Wayner, 2002).

Some literature such as Silman (2001) also refer to steganalysis as the destruction of the hidden information. And it can be done even if there is no knowledge of the existence of the hidden information (Krenn, 2004). However, forensics is about finding information and not destroying it. But it may indeed be reasonable to do so in other contexts. This is mainly because recovering hidden information can become rather complex and sometimes impossible without knowing which tool or technique was used for the steganography (Johnson & Jajodia, 1998). Even if the steganographic tool was somehow discovered, extracting the hidden information can prove to be rather daunting as most algorithms employ cryptographic techniques to scramble the secret message when it is embedded (Wayner, 2002). Although it use to be possible in classical steganographic systems where the security lies in the secrecy of the encoding scheme (Provos & Honeyman, 2002), modern systems have adopted Kerchoff's principle of cryptography, and the security depends on the secret key that is used to encode and not the encoding scheme (Provos & Honeyman, 2002; Krenn, 2004).

Therefore the challenge of recovering the hidden information remains for the forensic investigator, but the first step would be to identify suspicious articles with hidden information. According to Silman (2001) steganalysis attacks depend on the information that is available to the steganalyst such as:

- when only the steganographic object is available
- when the steganographic algorithm is known and the steganographic object is available
- when the steganographic object and the original cover object is available
- when both the steganographic and the cover object is available and the steganographic algorithm is known

If both the steganographic object and the cover object is available, checking and comparing file attributes such as size, file format, last modified timestamps, and colour palette can give some clues whether some information has been hidden (Krenn, 2004). However in forensic investigations, the most likely situation the investigator will be in is when only the steganographic object is available, and that is assuming if an object can be classified as a steganographic object in the first place. In such a situation it would not be possible to make comparisons with attributes of the original cover image and the steganographic image such as size as mentioned by (Silman, 2001).

Steganographic systems generally leave detectable traces (Provos & Honeyman, 2002). This is because often the process alters media properties and introduces degradations or abnormal characteristics that can be used as steganographic signatures for detection (Johnson & Jajodia, 1998). Although these cannot be detected by the human eye due to careful application of steganography, the signatures left can be electronically discovered (Silman, 2001). These signatures can also be used to identify the steganographic tools and techniques (Johnson & Jajodia, 1998), thereby aiding the investigator in the retrieval of the hidden information.

A commonly used steganographic technique is to make use of the LSB data, because the LSB data mostly appears random to a human observer although it contains hidden patterns (Wayner, 2002). Statistical analysis of the LSB data is a widely used method for detecting these patterns (Krenn, 2004). One of the most common pattern is a correlation between the High-Order Bits and the LSB which is often introduced by the hardware, such as the camera, used to generate the original data (Wayner, 2002). This attack is mostly successful because most of the steganographic algorithms operate under the assumption that the LSB is random, however statistical analysis can detect changes made to the LSB especially in the case of encrypted messages as it is more random and has higher entropy (Krenn, 2004).

Comparisons and analysis of numerous original and steganographic images can reveal anomalies and patterns can be classified. These can be detected due to factors such as unusual sorting of colour palettes, relationship between colours in colour indexes, and in exaggerated noise (Johnson & Jajodia, 1998).

LIMITATIONS IN STEGANALYSIS

Although there are some techniques that can detect steganography there are major problems that steganalysts face. Even if there are noticeable distortions and noise, predictable patterns cannot always be detected. Some steganographic techniques are particularly difficult to detect without the original image (Johnson & Jajodia, 1998). And in most cases, it is highly unlikely that a forensic investigator will be conveniently presented with the steganographic and original image.

To avoid detection, some steganographic technique spread the information and the diffusion makes it harder and less suspicious for detection. Some steganographic tools even use Random Number Generators (RNG) to make the LSB choosing process more random and to distribute the distortions throughout the file (Wayner, 2002)

According to Wayner (2002) another approach in defending against statistical attacks is not to saturate the cover image by packing in too much data, thereby leaving most of the LSB untouched hence making it highly indistinguishable from an untouched pure file.

Even until today, most steganalysis techniques are based on visual attacks and methods beyond this are being explored. Unfortunately a general steganalysis technique has not been devised (Johnson & Jajodia, 1998). While visual attacks are more prominent, JPEG images, which is one of the most commonly distributed type of image format, the steganographic modifications take place in the frequency domain. This means that this type of steganography is not susceptible to visual attacks unlike in image formats such as GIF images where the modifications happen in the spatial domain (Provos & Honeyman, 2002).

In order to verify the claims about terrorists using the Internet to distribute secrets using steganography, Niels Provos created a cluster that scans images from newsgroups to detect steganographic content (Krenn, 2004). In Provos' and Honeyman's (2002) work, upon investigating two million images from particular sources in the Internet, they were unable to find a genuine message and suggest the following explanations:

- steganography is not significantly used on the Internet
- the sources of the images analysed are not used for steganographic communication
- the steganographic systems detectable by the study are not being used
- strong passwords, not susceptible to dictionary attacks have been used by all steganographic systems users.

For reasons that no hidden messages were discovered, it raises the question of the practicality of such detection systems (Krenn, 2004).

EVALUATION OF STEGANALYSIS TOOLS

In order to evaluate the steganalysis tools, it is essential that the whole process is forensically sound to ensure the validity of the findings. Therefore, the following are the steps that will be followed throughout the process:

1. obtain the steganographic and steganalysis tools
2. verify the tools (to ensure the tools is doing what it claims)
3. obtain cover images, and generate MD5 hashes
4. apply steganalysis on cover images, and generate MD5 hashes
5. generate steganographic images, and generate MD5 hashes
6. apply steganalysis on the steganographic image, and generate MD5 hashes

In each of the steps where the cover images or the steganographic images are involved, MD5 hashes have been used to verify whether the image has changed in any sense.

Obtaining the tools

To keep the evaluation as realistic as possible, and the circumstances applicable to a wide range of users, the steganographic tools have been chosen based on how easy it is to obtain it, and the type of images it deals with. The tools that will be used are Steghide (Steghide Website, 2003), Outguess (Provos, 2004), and Digital Invisible Ink Toolkit (DIIT) (Hempstalk, 2005). All these tools are freely available, including their source codes, and can be downloaded by anyone from the Internet. Which means that anyone with programming experience can even make changes to the steganographic algorithms used. They can also be used on both Windows and Linux platforms. Therefore, this covers a wide range of users, who can make use of these tools. The following table (table 1) gives their version information and the output steganographic image format.

Table 1: Steganographic Tools

Tool	Version	Output Image Format	Platform	Source
Steghide	0.5.1-8	JPEG, BMP	Windows & Linux	(Steghide Website, 2003)
Outguess	1:0.2-6	JPEG, PNM	Windows & Linux	(Provos, 2004)
Digital Invisible Ink Toolkit (DIIT)	1.5	PNG, BMP	Windows, Linux & Mac OS	(Hempstalk, 2005)

Steghide can be used to hide data in JPEG and BMP image formats. It uses a graph-theoretic approach to perform the steganography (Steghide Website, 2003). More detailed descriptions of the tool can be found in the documentations available on Steghide Website (2003). Outguess performs steganography by inserting the hidden information into the redundant bits of the cover image. Its steganographic technique is able to protect the steganographic JPEG image from statistical attacks based on frequency counts (Provos, 2004). DIIT is a steganographic tool that allows to use four highly customisable algorithms to perform the steganography. The algorithms are BlindHide, HideSeek, FilterFirst, and BattleSteg (Hempstalk, 2005).

Similar to the steganographic tools, the choice of steganalysis tools were made based on its availability as a free software, and also its approach. It is important to note that there are other more expensive commercial steganalysis tools, such as StegAnalyzerSS (Steganography Analyzer Signature Scanner) by Backbone Security (2007). According to Backbone Security (2007), StegAnalyzerSS scans for unique hexadecimal byte patterns or known signature patterns in order to detect the steganography. But for this evaluation, two freely available tools were chosen. The first one is Stegdetect developed by Niels Provos and it can detect jsteg, jphide, invisible secrets, outguess 01.3b, F5 (header analysis), appendiX and camouflage steganographic schemes (Provos, 2004). This tool will be used on JPEG images generated by Steghide and Outguess. The next steganalysis tool was developed by Guillermito to implement the chi-square analysis to perform a statistical attack and detect any hidden messages (Guillermito, 2004). It will be used on the BMP images generated by Steghide and DIIT.

Table 2: Steganalysis Tools

Tool	Version	Steganalysis Attack
Stegdetect	0.6	JPEG image by Steghide and Outguess
Chi-Square	0.1	BMP image by Steghide and DIIT

The machine used to do the evaluation is based on Debian Linux. Therefore all the mentioned steganographic tools and steganalysis tools were obtained using the Debian repositories available on the Internet accept for the DIIT and Chi-Square. DIIT is provided as a JAR package from the website and Chi-Square is provided in EXE form compiled to run on Windows platform. However it was possible to use Wine to run the EXE file in the Linux environment.

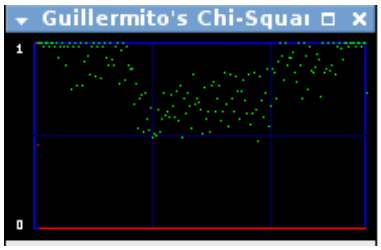
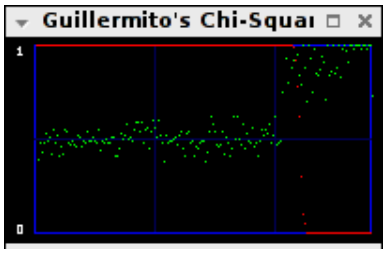
Verifying the tools

In order to verify that the tools are doing what they claim they can do, it has to be verified using test data. The steganographic tools test, verifies that the tools are able to hide a secret message in a cover image, and is able to retrieve the exact message using that tool, which would confirm the steganographic process.

In order to verify the steganographic tools, a sample text file was first created. The MD5 of the files was generated and recorded for later reference. All three steganographic tools were used to hide this text file in image files, and these tools were used to retrieve the hidden text file. The retrieved text files from each tool were used to generate their MD5 hashes, and was compared with the first MD5 hash that was generated. The results showed that they are all identical and was able to retrieve the exact data, that was hidden in the first place.

The steganalysis tools test, verifies whether it can detect the presence of a hidden message, using test steganographic images. The test steganographic images to test Stegdetect was obtained from the Stegdetect 0.6 source package available from the (Provos, 2004), and for the Chi-Square test, the images were obtained from (Guillermito, 2004). The source of each test steganographic image is the same source as the actual steganographic tool, therefore this guarantees the authenticity of the test images.

Table 3: Steganalysis Tools Verification Tests

#	Image	MD5 hash	Output
1	 (testimg.jpg found in http://www.outguess.org/stegdetect-0.6.tar.gz)	**before:** `01a77444369f4de7c7e3aea597f30324` **after:** `01a77444369f4de7c7e3aea597f30324`	`$ stegdetect testimg.jpg testimg.jpg : jphide(***)`
2	 (http://www.guillermito2.net/stegano/tools/googlemondrian.bmp)	**before:** `c4d2fc1028910ba53841ddaeb435d05e` **after:** `c4d2fc1028910ba53841ddaeb435d05e`	
3	 (http://www.guillermito2.net/stegano/tools/googlemondrian_02k.bmp)	**before:** `b2b15002f0b23b741c84c0bb0fdf53f7` **after:** `b2b15002f0b23b741c84c0bb0fdf53f7`	

In the above table (table 3), the first entry shows the Stegdetect test on the image testimg.jpg, which was obtained from the stegdetect-0.6.tar.gz source package and the test shows that jphide was used to embed a secret in it. The second and third test shows the Chi-Square test. Since the output is in the form of a graph, the test was carried out on a plain image with nothing embedded in it, which is the second entry. The third entry is the same image with 2KB of data embedded in it. According to Guillermito (2004), the red line (bottom line in entry 2 output) is the result of the chi-square test and if it is close to 1, then it shows that there is a high probability of an embedded message. If the green curve (collection of dots) is close to 0.5, then again it shows that there is a random message embedded. And lastly, the vertical blue lines indicate intervals of 1KB. As it can be seen, In entry 2, the red line is on 0 (zero) and the green curve is spread out. In entry 3, the red line is on 1 until the 2KB interval and the green curve is also close to 0.5 until the 2KB interval. These tests verify that both the steganalysis tools are working.

Obtain cover image

The cover image was first acquired using a Digital Camera. The image was originally in JPEG format in 680x480 resolution. Since a BMP image was also required for the evaluation, a second image in BMP format was generated using the same JPEG image using Gimp. Once both the cover images have been obtained, the MD5 hashes for both the images were created. As it can be seen in the following table (table 4), they are different, because they use different compression algorithms.

Table 4: Cover Images

flower.jpg	**flower.bmp**
	
MD5: `f34cc0ae3fb2a1c9be2faa674a2812d0`	**MD5**: `de24e73fd06702f577495af16eea7ddb`

Steganalysis of cover image

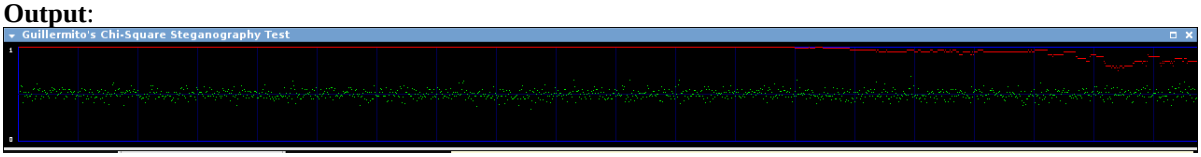
Steganalysis is done on the cover images, to ensure that there are no hidden messages embedded in the first place. Even though, in this particular case, there is knowledge that there are no messages hidden, it is a necessary step to make the process forensically sound.

Table 5: Stegdetect Test on flower.jpg

MD5 after test: `f34cc0ae3fb2a1c9be2faa674a2812d0`

Output:
`$ stegdetect flower.jpg`
`flower.jpg : negative`

Table 6: Chi-Square Test on flower.bmp

MD5 after test: `de24e73fd06702f577495af16eea7ddb`
Output: 

As seen in table 5, the Stegdetect test is negative for any embedded messages. However the Chi-Square test output shows all the characteristics of a graph that would show high probability of random embedded message. This is definitely a false positive, therefore it would be pointless to continue with this BMP image. The output might have been because the original BMP image contains too much random data. Another BMP image is needed and to avoid the same situation, the BMP image was created from scratch using Gimp.

As it can be seen in table 7, the new BMP image constructed from scratch satisfies the test as a clean image with no embedded messages. Although it maybe unlikely that an actual user would go through this process of creating an image from scratch in order to hide a message, this has been done for the sake of this evaluation to proceed by using a valid cover image, with respect to the Chi-Square test.

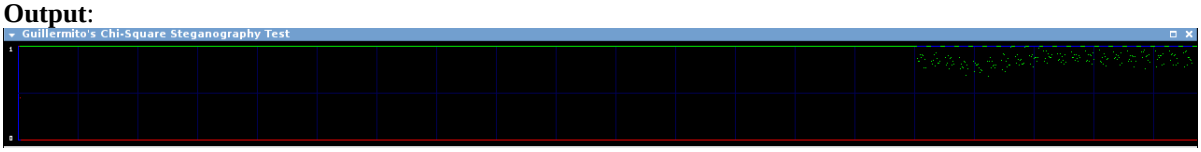
Once the steganalysis was carried out on the cover images, MD5 hashes were generated for each image and compared with the original hashes. The comparisons reveal that the steganalysis process did not alter the image as the hashes match.

Table 7: New BMP cover image

newbmp.bmp

MD5 before test: `9b190be2345100aebad2493e0d915522`

Output:



MD5 after test: `9b190be2345100aebad2493e0d915522`

Generate steganographic image

Each image was embedded with and without passwords for the encryption of the hidden message. The hidden message in the following cases are text files of different sizes. Different sizes have been used due to the different input requirements of the steganographic tools. Once the steganographic image was created, MD5 hashes of each image reveal that they have indeed been altered by the steganographic process.

```
$ steghide embed -cf flower.jpg -ef msg_small.txt -sf
steghide_np_flower.jpg
Enter passphrase:
Re-Enter passphrase:
embedding "msg2.txt" in "flower.jpg"... done
writing stego file "steghide_np_flower.jpg"... done
```

Figure 1: Steghide Process for JPEG images

The above figure (figure 1) shows the process of creating a steganographic image “steghide_np_flower.jpg” by using Steghide, using the cover image “flower.jpg”, and secret message in “msg_small.txt” file. No passphrases were entered. Another steganographic image was created by using a passphrase with the above process, to generate “steghide_wp_flower.jpg”. The following figure (figure 2) shows the Steghide process for generating the BMP file. Similar to the previous method, “steghide_np_newbmp.bmp” was generate without a passphrase, where as “steghide_wp_newbmp.bmp” was generated using a passphrase.

```
$ steghide embed -cf newbmp.bmp -ef msg_big.txt -sf
outguess_np_newbmp.bmp
Enter passphrase:
Re-Enter passphrase:
embedding "msg_big.txt" in "newbmp.bmp"... done
writing stego file "steghide_np_newbmp.bmp"... done
```

Figure 2: Steghide Process for BMP images

The following (figure 3) is the process of generating the steganographic image using Outguess. The same cycle was followed by generating with and without passphrases as “outguess_wp_flower.jpg” and “outguess_np_flower.jpg” respectively.

```
Reading flower.jpg....
JPEG compression quality set to 75
Extracting usable bits:   36976 bits
Correctable message size: 12962 bits, 35.06%
Encoded 'msg_smallest.txt': 7912 bits, 989 bytes
Finding best embedding...
    0:  3916(49.3%)[49.5%], bias  4169(1.06), saved:     5,
total: 10.59%
    5:  3876(48.8%)[49.0%], bias  4125(1.06), saved:    10,
```

```
total: 10.48%
   30:  3882(48.9%)[49.1%], bias  4113(1.06), saved:     9,
total: 10.50%
   47:  3898(49.1%)[49.3%], bias  4083(1.05), saved:     7,
total: 10.54%
   56:  3901(49.1%)[49.3%], bias  4048(1.04), saved:     6,
total: 10.55%
   99:  3883(48.9%)[49.1%], bias  3981(1.03), saved:     9,
total: 10.50%
99, 7864: Embedding data: 7912 in 36976
Bits embedded: 7944, changed: 3883(48.9%)[49.1%], bias: 3981,
tot: 36927, skip: 28983
Foiling statistics: corrections: 1788, failed: 3, offset:
92.138055 +- 203.759058
Total bits changed: 7864 (change 3883 + bias 3981)
Storing bitmap into data...
Writing outguess_np_flower.jpg....
```

Figure 3: Outguess process

The next tool that is used is DIIT. The following is a screenshot (figure 4) of the GUI of the tool. As it can be seen, it also provides an option to enter a password for encryption. Like the previous processes, in this process also, steganographic images were created with and without passwords as “diit_wp_newbmp.bmp” and “diit_np_newbmp.bmp” respectively.

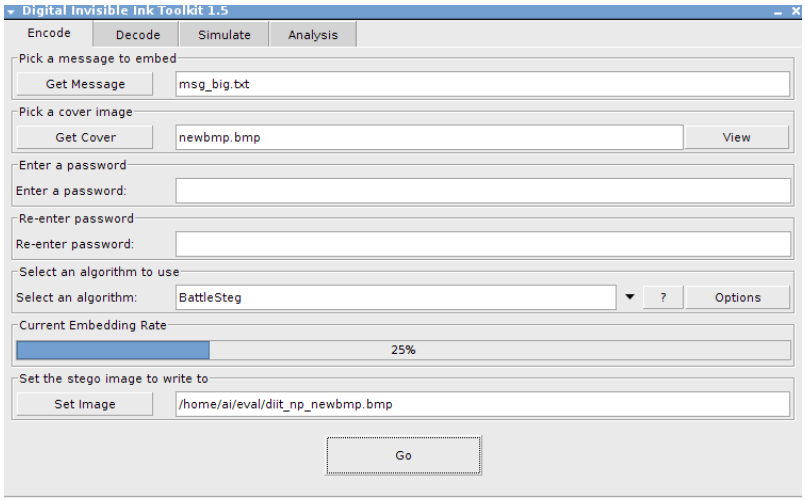


Figure 4: DIIT Screenshot

After each process, the MD5 hash was generated for original and the steganographic images. The following figure (figure 5) shows the MD5 hashes for all the images involved the Steghide, Outguess, and DIIT steganographic processes.

```
f34cc0ae3fb2a1c9be2faa674a2812d0  flower.jpg

4c2a9fb3860b299460a4be912a806437  steghide_np_flower.jpg
cdac07608cdf45f1e62ab96086dc362e  steghide_wp_flower.jpg

e7cd6d440badb0404db9e02f1c2dd9c6  outguess_np_flower.jpg
bbd68076246b513669e94180ee02ee5b  outguess_wp_flower.jpg

9b190be2345100aebad2493e0d915522  newbmp.bmp

54b03c5c48e374f697abb2809c4d3222  steghide_np_newbmp.bmp
a463186d7cbc0bfc1f1af13f2117c016  steghide_wp_newbmp.bmp

01f4acd389266a01a07acba0153108a6  diit_np_newbmp.bmp
fc914686e06b46e5672a1fdaea72c235  diit_wp_newbmp.bmp
```

Figure 5: MD5 hashes

As it can be seen, the original images (flower.jpg and newbmp.bmp) were not altered during the steganographic process and each of the resulting steganographic image is different from each other.

Steganalysis of steganographic images

The steganalysis is carried out using Stegdetect and Chi-Square. The following (figure 6) shows the result of Stegdetect on all the JPEG images. Once Stegdetect was carried out, MD5 hashes were generated for each JPEG image, but revealed that there were no changes. The result of the Stegdetect shows that, it was unable to detect the steganography of Steghide and Outguess.

```
outguess_np_flower.jpg : negative
outguess_wp_flower.jpg : negative
steghide_np_flower.jpg : negative
steghide_wp_flower.jpg : negative
```

Figure 6: Stegdetect Results

The following (figure 7-10) are the results of Chi-Square analysis on the BMP images. As it can be seen, the results cannot confirm the presence of a hidden message.

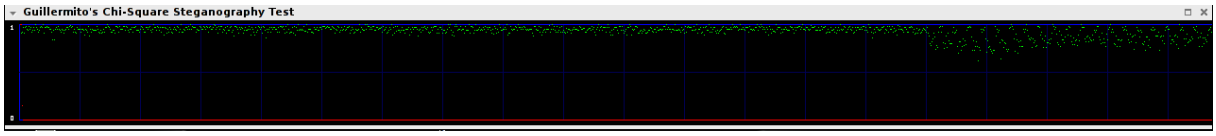


Figure 7: Chi-Square Result for Steghide (no passphrase)

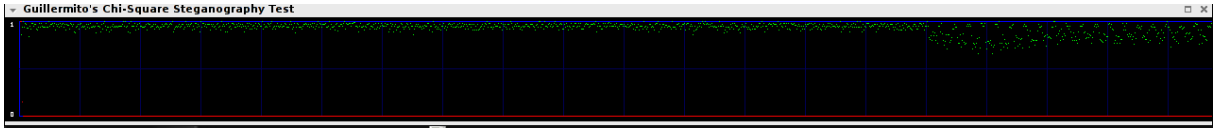


Figure 8: Chi-Square Result for Steghide (with passphrase)

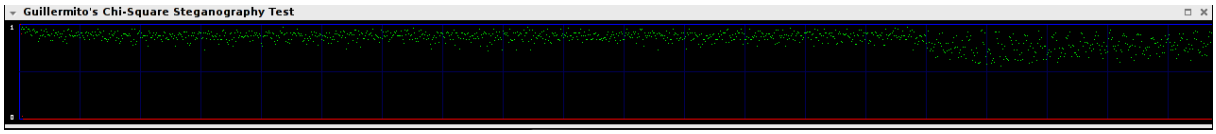


Figure 9: Chi-Square Result for DIIT (no passphrase)

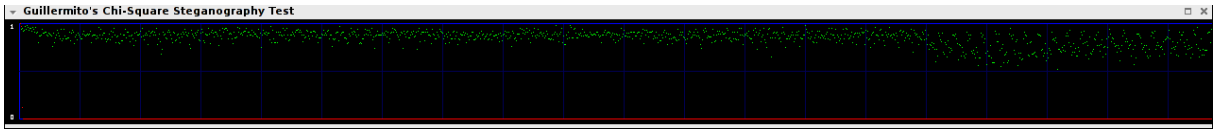


Figure 10: Chi-Square Result for DIIT (with passphrase)

None of the tools were able to positively identify the existence of hidden information. The messages were embedded in plain and encrypted form. The MD5 hashes show that the resulting images are different when the encryption is used by providing the passphrase during the steganographic process. However, the steganalysis was unable to recognise the existence of the hidden messages.

CONCLUSION

From the information that has been presented in this paper, it would be difficult to come to a firm conclusion regarding the state of steganalysis tools. Since it is not an extensive research with large amounts of data sets, it would be arguable if such a conclusion is made. However, it can be said that steganalysis is not as straight

forward or convenient as steganography. This translates to a great deal of advantage for those who hide secrets using steganography. And a huge disadvantage for the forensic analysts, who has the challenge of detecting and retrieving the hidden messages without destroying it.

Furthermore, it is also apparent that steganalysis fails when such tools are applied to detect steganographic techniques it wasn't designed to detect. It has also been observed that, false positives are also possible when generic techniques are used to detect factors such as randomness of LSB. Perhaps with more data and research, these tools can be enhanced to be more effective and accurate.

As steganographic tools are easily available in different varieties for anyone who intend to keep or communicate secrets, and with the emerging signs of its use in different arenas, forensic analysts face new challenges in their investigations. Criminals would indeed exploit every opportunity available to ensure the success of their plans. This could involve mass distribution of terror plans over the Internet or even more covert means of transmitting and storing illegal content on a portable storage devices.

Whatever the case maybe, it cannot be denied that there is a need to be concerned about the current state of forensic knowledge and tools available in this particular area of computer science. Perhaps it is because of a lack of interest among academics and other stakeholders due to less encouraging results of current research. Or it maybe because there is an unrealistic expectation of a magic pill to this problem. Nevertheless, as Johnson and Jajodia (1998) has mentioned, developments in steganalysis techniques will be extremely useful for law enforcement authorities in computer forensics, and an urgently needed development.

REFERENCES

Backbone Security (2007). *SARC – steganography analysis and research center*. Retrieved October 7, 2007, from http://www.sarc-wv.com/stegalyzerss.aspx

Guillermito (2004). *Steganography: a few tools to discover hidden data*. Retrieved September 29, 2007, from http://www.guillermito2.net/stegano/tools/index.html

Hempstalk, K. (2005). *Digital Invisible Ink Toolkit*. Retrieved September 28, 2007, from http://diit.sourceforge.net

Johnson, N. F., & Jajodia, S. (1998). Steganalysis: the investigation of hidden information. *Proceedings of the 1998 IEEE Information Technology Conference.* (pp. 113-116). Syracuse, New York, USA.

Kelley, J. (2001a). *Terrorist instructions hidden online*. Retrieved September 14, 2007, from http://www.usatoday.com/tech/news/2001-02-05-binladen-side.htm

Kelley, J. (2001b). *Terror groups hide behind web encryption*. Retrieved September 14, 2007, from http://www.usatoday.com/tech/news/2001-02-05-binladen.htm

Krenn, R. (2004). *Steganography and steganalysis*. Retrieved September 8, 2007, from http://www.krenn.nl/univ/cry/steg/article.pdf

McCullagh, D. (2001). *Secret messages come in .wavs*. Retrieved September 14, 2007, from http://www.wired.com/politics/law/news/2001/02/41861

Phadnis, S. P. (2007). *Data leak: cyber sherlocks outwit hackers*. Retrieved October 13, 2007, from http://economictimes.indiatimes.com/Infotech/Data_leak_Cyber_sherlocks_outwit_hackers/articleshow/2451089.cms

Pieprzyk, J., Hardjono, T., & Seberry, J. (2003). Fundamentals of computer security. Berlin: Springer.

Provos, N., & Honeyman, P. (2002). *Detecting steganographic content on the internet*. Retrieved September 2, 2007, from http://www.citi.umich.edu/u/provos/papers/detecting.pdf

Provos, N. (2004). *OutGuess - universal steganography*. Retrieved September 30, 2007, from http://www.outguess.org

Schneier, B. (2000). Secrets & lies: digital security in a networked world. Indianapolis, Indiana: Wiley Publishing, Inc.

Silman, J. (2001). *Steganography and steganalysis: an overview.* Retrieved September, 8, 2007, from http://www.sans.org/reading_room/whitepapers/stenganography/553.php

Steghide Website (2003). *Steghide*. Retrieved September 28, 2007, from http://steghide.sourceforge.net

Wayner, P. (2002). Disappearing cryptography information hiding: steganography & watermarking (2nd ed.). Amsterdam: Morgan Kaufmann Publishers.

COPYRIGHT