STEGANOGRAPHY AND THE FLAWLESS CARRIAGE OF SENSITIVE DATA

PRAMNEET KAUR

Department of Information Systems, Netaji Subhash Institute of Technology, Dwarka, New Delhi, India
*Corresponding Author: Email- pramneet@gmail.com

Abstract- Steganography allows for the concealment of files and messages within other files. This allows for the easy transportation of sensitive data without prying eyes being able to tell a message is being transported. By using steganography techniques, observers cannot tell the difference between encoded images and their originals, as shown in the following research. This leads to the suggestion that steganography is an excellent tool for the transport of secure data over the Internet, or by physical media – especially since the original encoding media that the data was encoded into is rarely available for comparisons.
Keywords- Steganography, Secure Data Transport, Data Hiding.

Citation: Pramneet Kaur (2012) Steganography and the Flawless Carriage of Sensitive Data. Journal of Information Systems and Communication ISSN: 0976-8742, E-ISSN: 0976-8750, Volume 3, Issue 1, pp- 64-67.

Introduction

Steganography is the hiding of a message within another media. The purpose of this paper is to show how steganography can be effectively used to transport sensitive data over the Internet in a secure fashion. A prime candidate for steganography is the use of an image to conceal a hidden ciphered message. Steganography improves encryption and security by creating a medium in which sensitive data can be passed through prying eyes via a file, without alerting anyone that the transmitted file actually contains a message. Steganography should be implemented more in practice today as cyber-terrorism and data-theft becomes an everyday occurrence. Steganography is a relatively old technology, but is still very young in regards to modern usage. With the high security concerns of corporations and individual users alike, the secure transportation of data needs to be a viable option. There are numerous techniques and methods for steganography, which have been highly researched and used in practice. This paper shows how well steganography performs in real-world applications in regards to the transportation of sensitive data. The following research uses encoded images and sound files, which were pre-sented to test subjects in pairs (one encoded file and its original). The subjects were then asked to analyze them and attempt to pick out the steganographic media from each pair. This study shows that, even when the original non-encoded file is available for comparison, steganographic files are not discernable by standard observation. This paper also shows how modern steganography detection techniques perform when put to the test.

Background
What is Steganography?

"The concept of hiding information in other content has existed for centuries; the formal study of information hiding is called steganography." [1] Steganography is the practical science of hiding information inside other media with the intention of giving the impression that no hidden data is present. Steganography is not a new technology, having been practiced for thousands of years dating back to early mapmakers. Steganography allows a sender to embed a hidden file or message inside a cover file. A cover file is simply a file that is used to embed hidden data into. This cover file may be a graphics image, an audio file (such as a WAV or

MP3 file), or even a binary executable. Cryptography has the goal of preventing the viewing of sensitive data by obfuscating the message so only the sender and recipient can view it. Steganography is intended to take cryptography to the next level by attempting to prevent the impression of the existence of any sensitive data.

Steganography's main goal is to avoid detection; to deny the existence of sensitive data inside the cover file. In the use of steganography, a cover file and hidden file are used. It is assumed that any eavesdroppers will have no access to the original cover file in question. Steganography techniques try to change the original cover file as little as possible in terms of quality and file size, in order to create the strongest security environment possible. "In steganographic applications there are two levels of security. The first is not allowing an observer to detect the presence of a secret message. The other is not allowing the attacker to read the original plain message after detecting the presence of secret information."[2]

Common media for steganography includes images (e.g.: JPEG, GIF, BITMAP, PNG), audio files (e.g.: MP3 and WAV), and even executable binaries (e.g.: EXE executables). Just about any file type that has slack or white space in it can be used for steganography – however images are usually the typical medium used for steganography purposes.

Uses of Steganography

Steganography has a wide array of uses. For example, it can be used for digital watermarking, e-commerce, and the transport of sensitive data. Digital watermarking involves embedding hidden watermarks, or identification tokens, into an image or file to show ownership. This is useful for copyrighting digital files that can be duplicated exactly with today's technologies.

E-commerce allows for an interesting use of steganography. In current e-commerce transactions, most users are protected by a username and password, with no real method of verifying that the user is the actual card holder. Biometric finger print scanning, combined with unique session IDs embedded into the fingerprint images via steganography, allow for a very secure option to open e-commerce transaction verification. [3]
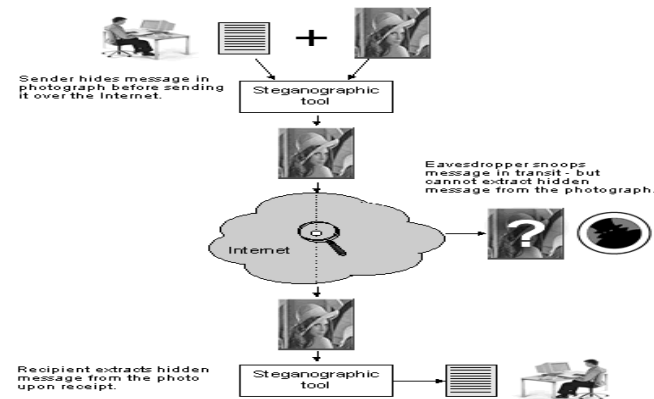
Fig. 1- Steganography on the Internet

The transportation of sensitive data is another key use of steganography. A potential problem with cryptography is that eavesdroppers know they have an encrypted message when they see one. Steganography allows (or tries to allow) the transport of sensitive data past eavesdroppers (see Figure 1) – without them

knowing any sensitive data has passed them. The idea of using steganography in data transportation can be applied to just about any data transportation method, from E-Mail to images on Internet websites. With proper steganography techniques applied, sensitive data can be placed on public systems where only the designated recipient knows where the message is located. For example, an auction on eBay.com could be used to place a hidden, steganographic message for a specified recipient to view.

However, no other browsers would have any idea the image contained a hidden message.

Implementation

Steganography can be implemented in various ways. "There are many different kinds of steganography, but all are based on finding unused space on paper, in sound, or in files in which to hide a message." [5] Many algorithms have been developed to provide robust and secure steganography – each of which uses different embedding techniques. A common technique is Least-Significant-Bit (LSB) embedding. This technique hides data bits in the last two significant bits of an image pixel. For example, a 500x500 pixel image has 250,000 pixels. If a small 249 character null-terminated ASCII message is embedded, we would need approximate 1992 bits (249 * 8 bits per character) for storage of the string. By breaking up the bit pattern for each character into pairs, we would need 4 pixels per character to store the message by storing 2 bits per pixel. This means only 1000 pixels ((249+1)*4) would need changing in their LSB (from out of a total 250,000). The "+1" in the above calculation comes from the null character used to terminate the string. This has very little effect on the overall image. [6]

Figure 2 shows the steganography process of the cover image being passed into the embedding function with the message to encode – resulting in a steganographic image containing the hidden message. A key is often used to protect the hidden message. This key is usually a password used by the decoding software to unlock the hidden message. Most steganography tools offer encryption of the hidden message before the embedding function is executed, so this key is also used to encrypt and decrypt the message before and after the embedding process.

More extensive and robust techniques are available for review, but they are beyond the scope of our intent in this paper. Many tools are available for the steganography of various media, including binary executables and MP3 audio files. In this paper we use the tools S-Tools and JPHIDE. Each of these tools is further discussed in the Methodology section of this paper. [4,9]
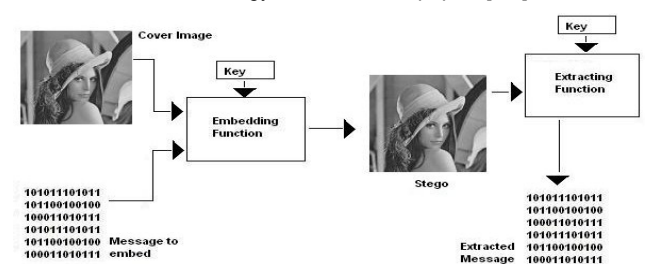
Fig. 2- The steganography process [10]

Detection

The detection of steganography is a rough science, at best. Some tools are present that attempt to detect the presence of a hidden

file inside other files. One such tool is StegDetect, written by Niels Provos. StegDetect attempts to detect steganographic content created by a handful of common tools. These tools include JPHIDE, OutGuess, Invisible, and JSteg.[9] A common method of detection is to analyze the least-significant bits of an image. Some steganography tools (like JPHIDE) leave a noticeable signature in the histogram of an image by using all of the first available LSBs to encode the hidden file (as opposed to a random selection or even distribution). StegDetect analyzes the histogram of a selected JPEG image to see if it has a signature created by being altered with steganography tools. This can be a daunting task – especially if the hidden file is very small relative to the overall cover file size (as discussed earlier in the brief LSB technique overview). The smaller the message that is hidden, the less impact there is on the cover file.

In a research paper written by Niels Provos, the author of StegDetect, an attempt was made to crawl images on eBay.com and scan them for steganographic content. Out of over two million images crawled and scanned, StegDetect found around 17,000 images supposedly altered with steganography – 15,000 of which were supposedly altered by JPHIDE (written by Allan Latham). Out of these 15,000 images, no genuine hidden messages were found. Thus leading Provos to conclude that steganography was not being used actively on the Internet - or that other tools (other than the ones detected by StegDetect) were being used to perform the steganography. [7]

Purpose

The purpose of this paper is to show that steganography is a strong solution for the transportation of sensitive data. When encrypted messages are too obvious for transmission, a more subtle approach needs to be taken. Steganography creates such an approach by hiding the existence of any sensitive data in the medium being transferred. Our research shows that, even when presented with the original cover images, computer users are not able to identify the steganographic file from the original. We also show that the steganography detection tool, StegDetect, does not identify the hidden files.

Methodology

To test our hypothesis that steganography provides a secure method of transporting sensitive data, we encoded a set of images and WAV files with the steganography tools S-Tools and JPHIDE. S-Tools is written by Andy Brown and allows for the facilitation of image (BITMAP and GIF) and WAV file steganography. "S-Tools applies the LSB methods discussed before to both images and audio files."[8] S-Tools also allows encryption and decryption of the hidden file with several different encryption algorithms. JPHIDE is a steganography tool written by Allan Latham that provides for JPEG steganography. Both tools support password protection of the hidden file.

The image pairs used include GIF, JPEG, and BITMAP images. A pair of WAV files was also used to test steganography in audio media. Within each pair, one file was encoded with a hidden message – the same message was used for all encoding done in this experiment. The test message is a null terminated 1868 character text message. This would require 14944 bits of storage (1868 * 8 bits per character). Using LSB steganography, 7472 pixels of storage would be needed in the cover file (see section Implemen-

tation for the arithmetic behind this number). After the files were created, they were presented to test subjects that examined the media files for any variations that may show one of the files was enhanced or modified by steganography. They then marked which image was different, or altered, or they marked a third choice signifying they could not tell any difference between the two files. The survey was given to the test subjects in the format of an HTML page – which we decided suited this research well as images are commonly transferred over the web via web-pages. The test subjects were advised to use only visual and audio feedback to analyze the file pairs – file size should not be taken into account since steganographic files in the wild cannot be compared to their original cover file's file size.

Due to time constraints, we chose to present the steganography files to thirty-four (34) test subjects. Each test consisted of four image pairs. The pairs consisted of JPEG, BITMAP, and GIF image formats of varying sizes. The BITMAP was used to represent a larger size cover file and the GIF file presented the smallest file size in the test. Two WAV file pairs were also used – with one pair representing a small file size, and the other being a much larger file. We predicted beforehand that the files with larger file sizes would be hardest to detect any alterations in – whereas smaller files would be more susceptible to visual changes, as there are less storage bits to work with. A fourth image pair was added to each test in which both images were unchanged. This provided a bias indicator to ensure test subjects were simply not guessing, and being honest if they truly did not know an answer.

After the steganographic images were tested on our test subjects, we used a tool called StegDetect to attempt to detect JPEG steganography. StegDetect is designed to find hidden messages in JPEG images encoded with JPHIDE (and a few other steganography tools) so naturally a batch of encoded JPEG images (encoded by JPHIDE) were used. The steganographic images used in this detection test all used the same cover files (a 34KB JPEG image and a 174KB JPEG). Messages of various file lengths were encoded into the cover file each time to produce different steganographic images.

Results

The results of this paper are broken up into two aspects: the survey results, and the detection results.

Survey Results

The results, shown broken up by pairs, and arranged according to test subjects' answers. Pair 1 indicates the JPEG image pair (34KB image), Pair 2 indicates the JPEG used as a bias indicator, Pair 3 indicates the GIF image pair (7KB image), and Pair 4 indicates the BITMAP image par. Wav Pair 1 indicates the smaller WAV file used (31KB) and Wav Pair 2 indicates the larger WAV file (229KB).

The results are easily read (as seen in Graph 1) – steganographic images are hard to detect, even when the original is given for comparison. Even the low quality, small GIF file used was only selected correctly by nine viewers (out of 34). The much larger BITMAP pair only resulted in five correct responses. We postulated earlier that the larger files would lead to lower detection rates – and that proved valid. The large BITMAP image used (Pair 4) yielded the highest success rate amongst the image pairs. The larger WAV file (Wav Pair 2) yielded the highest success rate as

well. The smallest file (Pair 3) was easiest to be detected, given that the message that was hidden was approximately 1/7th the total file size. This also supported our earlier prediction. Our bias indicator proved to work very well – only 6 subjects incorrectly guessed that it was altered (instead of choosing the correct "Do not know" answer).
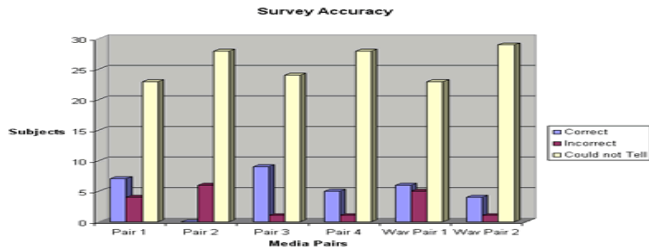

Fig.1- Graph version of survey results

After subjects answered the survey, they were asked how they evaluated the images. Unfortunately, a few subjects mentioned they use file size as an indicator – which we stated earlier should not be done. This shows that the success rates we have accumulated may have been much higher if these results were filtered out prior to analysis of the survey results. Other subjects who achieved some correct answers without "cheating" shared a common trait – they all worked in the graphics design industry. This leads us to believe that a trained eye could help detect steganography images – but only when they are able to be compared to the original cover file.

Overall, our findings support our hypothesis that steganography is a strong solution for the transfer of sensitive data. Even when the original cover file was available for comparison, steganographic images could hardly be determined from their original counterparts.

Detection Results

The detection results provided an interesting picture, which also supports our idea that bigger is better in regards to cover file size. Our steganography detection tool, *StegDetect*, managed to detect some steganographic images, but not many. We embedded various messages ranging from 220 characters to 6KB in the 34KB JPEG image. *StegDetect* was unable to detect any messages in the altered files with its default settings. When *StegDetect's* sensitivity was increased (a feature that allows *StegDetect* to be "more accurate") nearly all steganographic images were detected – minus the first 220-character file. This file could not be detected regardless of the sensitivity settings in *StegDetect*. We then tested *StegDetect* on various steganographic images built from the 174KB JPEG cover file. *StegDetect* enjoyed much less success on this test run – only one of the six-steganographic images generated was detected. Message sizes used in this run ranged from a 220-character text message to a 16KB image. The file detected was the steganographic image with a very large hidden message – the largest *JPHIDE* could hide within the cover file (the 16KB image). Regardless of sensitivity settings, *StegDetect* could not find the other steganographic images with hidden messages of varying sizes (including a file that was 11KB in size).

This leads us to believe that *StegDetect* works somewhat reliably on smaller images, but not on larger images – especially those with small hidden messages. To bypass *StegDetect*, only a small hidden message in a large cover file is needed. *StegDetect* only

supports JPEG images and four steganography encoding programs (all of which only use JPEG cover files). Given that *StegDetect* was the only testable software we could find, we conclude that steganography detection is very primitive in software form – and that it can be bypassed by simply using GIFS, BITMAPS, WAVS, and other media for steganography cover files.

Conclusion

Steganography is a powerful technique of hiding data in other files. This paper briefly covers the background, implementation, uses, and detection of steganography. There are many tools, like *S-Tools* and *JPHIDE*, that can be used by computer users to easily enhance their security with steganography. With steganography, sensitive data can be transported securely over the Internet and other mediums, without tipping off eavesdroppers and hackers.

My research, on a small number of test subjects (due to time restrictions) shows that steganography effectively encodes hidden messages in media files without the viewer being able to notice – even if they have the original cover file to compare it to. This research concludes that steganography is effective at hiding hidden messages without altering the cover file noticeably. My research also shows how unreliable and "young" steganography detection tools are. We were only able to find one worthy steganography detection tool to test (*StegDetect*) – and it did not perform well in our test. *StegDetect* showed marginally accurate detection in small images with large encoded hidden messages, but failed in nearly all tests on larger file sets.

After completing this research, we can choose to further extend our steganography research in a few different ways. Research could be performed to accurately find a plane of reliability with common steganography tools. In other words, we would determine exactly how large hidden messages can be in relation to their cover files without visually altering the cover file during the encoding process. Attention could be given to the improvement of steganography detection, as we have shown the programmatic detection of steganographic files is very weak. Lastly, we would like to enhance security aspects of steganography tools to counter any detection techniques that may be presented in the future. Now that we have a solid idea of how steganography is implemented (and sometimes detected) we could improve steganography algorithms even more.

References
[1] Acken J. (1998) *Communications of the ACM*, 41(7), 75-77.
[2] Alturki F. and Mersereau R. (2000) *ACM workshops on Multimedia,* 131-134.
[3] Bolle R., Connell J. and Ratha N. (2000) *Workshops on Multimedia,* 127-130.
[4] Brown A. (2003) *S-Tools 4*.
[5] Castelluccio M. (2001) *Strategic Finance* 83(5) 59.
[6] Chang L., Longdon G. and Moskowitz I. (2000) *Workshop on new security paradigms*, 41-50.
[7] Honeyman P., Provos N. (2001) *http://www.citi.umich.edu/techreports/reports/citi-tr-01-11.pdf*.
[8] Johnson N. (1995) *Steganography http://www.jjtc.com/stegdoc/steg1995.html*.
[9] Provos N. *Steganography detection with stegdetect. eganography*.
[10] Tran T. (2000) *Steganography: the art of hiding data*.