Steganography and Steganalysis

Waheed Qureshi

Table of Contents

1 Abstract ... 3
2 Introduction ... 3
3 Steganography, a history ... 3
4 Media Types ... 4
4.1 Computer File System ... 4
4.2 Transmission Protocol ... 5
4.2.1 Hiding information within an IP header ... 5
4.3 Encoding data in text documents ... 6
4.3.1 Open Space methods ... 6
4.3.2 Syntactic Methods ... 7
4.3.3 Semantic Methods ... 7
4.4 Data hiding in the audio ... 8
4.5 Data hiding in the graphic files ... 8
5 Steganalysis ... 11
5.1 Independent Research ... 12
6 Conclusion ... 15

Table 4.2.1-1 The IP Header ... 5
Table 4.2.1-2 Hiding data in the identification field ... 6

Figure 4-1: Cover Image and Uniform Pixel Area ... 9
Figure 4-2: Encoded Picture and Decoded Message ... 10
Figure 4-3: Comparison of cover image with stego in binary ... 11
Figure 5-1: RGB Color Axis and Luminance Axis ... 13
Figure 5-2: Zigzag Pattern ... 14
Figure 5-3: Block Diagram of JPEG image compression ... 14

1 Abstract

The purpose of this paper is to present an overview of steganography. A brief history of steganography is provided along with techniques that were used to hide information. While this document focuses on hidden information in digital images, other mediums will also be discussed to provide the reader with general understanding of many available techniques. Finally a number of techniques will be discussed to perform steganalysis.

2 Introduction

Steganography is usually confused with cryptography. While related in many aspects, steganography and cryptography are not the same. The existence of a cryptographic message can easily be noticed by a casual observer as the messages are scrambled to hide the original content. Whereas, steganography hides the original message in other innocent messages to protect the information from prying eyes. The existence of the hidden message is not obvious to an ordinary observer and cannot be detected by naked eye in most cases.

Perhaps the biggest advantage that steganography has over cryptography is the fact that transmission of secret information is non-observable. However, unlike cryptography, steganography requires a magnitude of overhead to hide small amount of information. Compression techniques along with encryption can be applied to make a steganography system more productive and secure. Most of the current steganography software applications use these techniques to provide a robust system. This is necessary as once a steganographic system is discovered it is rendered useless.

Digital watermarks play a role by placing information within digital media. This information may constitute registration of ownership for copyright or a means to locate an image that has been distributed. For example commercial applications are available to search the web for images that contain watermarks. Once these watermarked images are found their presence is reported back to the registered owner of the image [11]. This document will not explore digital watermarks any further. Interested readers are encouraged to consult links provided at the end of this document.

3 Steganography, a history

While steganography has received a tremendous attention recently, its application goes back to Greek times. According to Greek historian Herodotus, the famous Greek tyrant Histiaeus, while in prison, used unusual method to send message to his son-in-law. He shaved the head of a slave to tattoo a message on his scalp. Histiaeus then waited until the hair grew back on salve’s head prior to sending him off to his son-in-law [10].

Wax cover tablets were also used by ancient Greeks to hide messages. The sender of the message would scrape off the wax from the tablet to write the message on the wood.

Once the message was written the wooden tablets were waxed again to hide the existence of the message. [14]. Steganography evolved over time and became very popular during World War II (WWII), where invisible inks were used to write in-between the lines of an innocent message to hide the information.

Germans used null ciphers or unencrypted messages to hide information in text document. For example a German spy sent the following null cipher message during WWII. [14]

Apparently neutral's protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue affects pretext for embargo on by-products, ejecting suets and vegetable oils.

After decoding this message to extract the second letter in each word, the hidden message appeared to be as follows:

Pershing sails from NY June 1.

Methods other than invisible ink and null cipher were developed later on to pass sensitive information. Some of these methods are discussed in the next sections of this document. Throughout the history different media types have been used to hide information. With advancements in computer industry this number is only increasing. Some of the media type that I will cover in this paper includes: Computer file system, transmission protocols, audio files and images. While this documents discusses briefly all of these media types, details would be provided for systems that use images as cover.

4 Media Types

4.1 Computer File System

Where it stores normal data, a computer file system can also be used to hide information between innocent files. For example a hard drive while showing the visible partition to a computer user may contain hidden partitions that can carry hidden files inside them. For example sfspatch is a kernel patch, which introduces module support for the steganographic file on a Linux machine. Sfspatch employs encryption along with steganographic techniques to hide information on the disk so it is not visible to a casual user [3].

FAT 16 system on Microsoft Windows hosts allocate 32 kilobytes of disk space to each file. If the file size is only a few kilobytes, the rest of the space can be used to hide information.

4.2 Transmission Protocol

Transmission Control Protocol (TCP) and Internet Protocol (IP) are some of the few protocols that can be used to hide information inside certain header fields. Some TCP/IP fields are either changed or stripped off by packet filtering mechanisms or through fragment re-assembly. However, there are fields that are less likely to change or altered. These fields include: Identification field, Sequence Number field and Acknowledge Sequence Number field [16].

4.2.1 Hiding information within an IP header

Table 4.2.1-1 depicts an IP header.

<table>
<tr><td>00</td><td>01</td><td>02</td><td>03</td><td>04</td><td>05</td><td>06</td><td>07</td><td>08</td><td>09</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="8">TOS</td><td colspan="16">Total length</td></tr>
<tr><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment offset</td></tr>
<tr><td colspan="8">TTL</td><td colspan="8">Protocol</td><td colspan="16">Header checksum</td></tr>
<tr><td colspan="32">Source IP address</td></tr>
<tr><td colspan="32">Destination IP address</td></tr>
<tr><td colspan="24">Options</td><td colspan="8">Padding</td></tr>
</table>

Table 4.2.1-1 The IP Header

The Identification field within an IP header provides network devices with a unique number to identify packets that may require reassembly. As presented by Neil F. Johnson in INFS 762 class at George Mason University (GMU), replacing the identification field with the numerical ASCII representation of the character to be encoded provides an easy way to hide information within this field. In his example, Johnson selects an unsigned integer to be transmitted as the identification field. The ASCII value of this integer can be achieved by dividing the integer by 256.

At the transmitting end client host construct a packet to include the desired identification number along with source and destination address. In the following example I have chosen 18432, 18688, 17408 and 17664 as the four identification field values for the four IP packets. This process is depicted in Table 4.2.1-2.

Sending Host	Receiving Host
21:55:35.763195 IP www.source.com.80 > www.destination.com.1444: S 788196017:788196017(0) ack 2449684694 win 32640 (ttl 64, id 18432)	….(ttl 64, id 18432/256) ASCII 72 = H
21:55:34.754811 IP www.source.com.80 > www.destination.com.1442: S 1093146807:1093146807(0) ack 2449313332 win 64240 (ttl 64, id 18688)	….(ttl 64, id 18688/256) ASCII 73 = I
21:55:33.711164 IP www.source.com.80 > www.destination.com.1439: S 1092753896:1092753896(0) ack 2448919278 win 64240 (ttl 64, id 17408)	…..(ttl 64, id 18688/256) ASCII 68 = D
21:55:32.304123 IP www.source.com 80 > www.destination.co.1436: S 1092243953:1092243953(0) ack 2448416550 win 64240 (ttl 64, id 17664)	…..(ttl 64, id 18688/256) ASCII 69 = E

Table 4.2.1-2 Hiding data in the identification field

Once the ASCII value of the identification field is calculated at the destination, the decoded message is found to be the word "HIDE."

Similar techniques can also be used to encode information in the Sequence Number field of a TCP packet.

4.3 Encoding data in text documents

According to Bender et al [17] softcopy text is one of the most challenging places to hide data. One reason for this is the lack of redundant information in the text files. Bender et al discusses three different techniques of hiding information in text files. These methods are: Open space method, Syntactic methods and Semantic methods.

4.3.1 Open Space methods

There are couple ways to employ the open space in text files to encode the information. This method works because to a casual reader one extra space at the end of line or an extra space between two words does not prompt abnormality. However, open space methods are only useful with ASCII (American Standard Character Interchange) format [17]. Bender et al presents three methods to exploit the white space for encoding purposes.

Inter-sentence space method encodes a "0" by adding a single space after a period in English prose. Adding two spaces would encode a "1". This method works, but requires a large amount of data to hide only little information. Also many word processing tools automatically correct the spaces between sentences.

End-of-line space method exploits white space at the end of each line. Data is encoded using a predetermined number of spaces at the end of each line. For example two spaces will encode one bit, four spaces will encode two bits and eight spaces will encode three bits and so on [17]. This technique works better than the inter-sentence space method, because increasing the number of spaces can hide more data. One disadvantage of this technique is to lose the encoded information if only a hard copy of the data is provided.

Finally, right-justification of text can also be used to encode data within text files. Calculating and controlling the spaces between words encode data in innocent text files. One space between words represents a "0" and two spaces represent a "1". However, this approach makes it difficult to decode the data as it becomes impossible to distinguish a single innocent space from an encoded one. For this purpose Bender et al used Manchester coding to group the bits. Hence "01" is interpreted as "1" and "10" is interpreted as "0". Whereas "00" and "11" are considered the null bit strings.

4.3.2 Syntactic Methods

Syntactic methods as suggest by Bender et al, exploit the use of punctuation and structure of text to hide data without scientifically altering the meaning of the message. For example the two phrases *"bread, butter, and milk"* and *"bread, butter and milk"* are grammatically correct but differ in the use of comma. One can employ this structure alternatively in a text message to represent either a "1" if one method is used and to represent a "0" if the other method is employed.

4.3.3 Semantic Methods

Semantic methods assign two synonyms a primary or secondary value. These values are then translated into a binary "1" or "0". Bender et al use an example where word "big" is assigned a primary and "large" is assigned secondary. Therefore, decoding a message would translate the use of primary to be "1" and secondary to a "0". Bender et al mentions a potential problem with this approach where synonyms cannot be replaced as it may change the meaning or structure of the sentence. For example calling someone "cool" has a different meaning than calling him or her "chilly" [17].

Other methods of encoding information in text documents can also be found in paper written by J. Brassil, S. Low, N. Maxemchuck, and L.O'Graman. There paper titled "electronic marking and identification techniques to discourage document copying" discusses many interesting techniques to achieve this goal.

4.4 Data hiding in the audio

Bender et al consider data hiding in audio signals to be especially challenging. This is due to the fact that Human Auditory System (HAS) operates over a wide dynamic range. But as they discuss in their paper there are still some possibilities to exploit a few available "holes." Prior to encode data in audio files one needs not only to keep in mind the sensitivity of HAS, but also the fact that audio signal travels between encoding and decoding. It could either travel to a storage media or transmitted over a medium. When these audio files are represented digitally, sample quantization method and temporal sampling rate also become a critical factor. Some of the techniques suggested by Bender et al to hide data into audio files include low-bit encoding and Phase coding.

In low-bit encoding data is embedded by replacing the Least Significant Bit (LSB) of each sampling point by a coded binary string. This results in a large amount of data that can be encoded in a single audio file. For example if the ideal noiseless channel capacity is I Kbps then the bit rate will be 8 Kbps given an 8 kHz sampled sequence. While the simplest way to hide data in the audio files, low-bit encoding scheme can be destroyed by the channel noise and re-sampling [17].

Phase coding when it can be used has proven to be most effective coding techniques in terms of signal to noise ratio [17]. In this method the phase of the original audio signal is replaced with the reference phase of the data to be hidden. Bender et al discovered that a channel capacity of approximately 8 bps can be achieved by allocating 128 frequency slots per bit with a little background noise. Bender et al also discuss methods for improving audio signal encoding under different communication channels. However, focus of this document is on hidden information in graphic files and interested readers are encouraged to visit "Techniques for Data Hiding" by Bender et al. A hyperlink to this document is provided at the end of this document.

4.5 Data hiding in the graphic files

With advancement in computer technology the field of steganography has grown rapidly, specifically when it comes to image files. There are a number of easy to use tools available on the Internet to hide information in image files. Most of the tools that I downloaded are fairly simple to use and do not require a prior background or expertise in steganography. For the purposes of this document I limited myself to S-Tools and Jsteg as these are some of the widely used. While very user friendly, S-Tools and Jsteg perform complex operations in the background to accomplish the goal of hiding information. Selecting different encryption and compression techniques can further complicate this process.

However, for users with minimal knowledge of digital image format and coding techniques, these tools only become an interface. To better understand and appreciate some of the processes used by these tools some understanding of digital image processing becomes essential. This information also helps in performing steganalysis, the art of detecting the presence of hidden information. Without having much exposure to digital image process myself, I read many documents and searched for tools to better understand the overall process.

While searching, I found a number of interesting tools that helped me in understanding digital images, coding schemes and use of colors in digital images. Keeping the length requirement of this document in mind, I would only discuss one of such tools. Written by Steve Tenimoto and his team at the University of Washington, pixel calculator is a very interesting tool. Pixel calculator is not only helpful in understanding digital images and use of pixels, but provides a neat feature of achieving some very basic steganography. Again, due to the length requirements, I will try to keep the use of images to as minimal as possible.

Pixel calculator is equipped with two basic tools. A zooming tool is provided to learn the exact pixel value by zooming into an image until the pixel values are visible. A calculator tool is then used to change or modify pixel values. Learning the pixel values and changing them using the calculator is the key in hiding the information inside an image. Figure 4-1 depicts the image file I used as cover to hide the information.

Following the instruction provided at pixel calculator web site [4], I used the calculator tool to manipulate some pixel values. Since the calculator only works with 256 grayscale images, instructions led me to divide the whole image by integer value of "2." Next step was to multiply the whole image by integer value "2." Combined together both of these operations remove all of the odd numbered gray values in the image. I then used the zooming tool to find an area of interest where neighboring pixel values are close to each other. These values are visible in Figure 4-1.

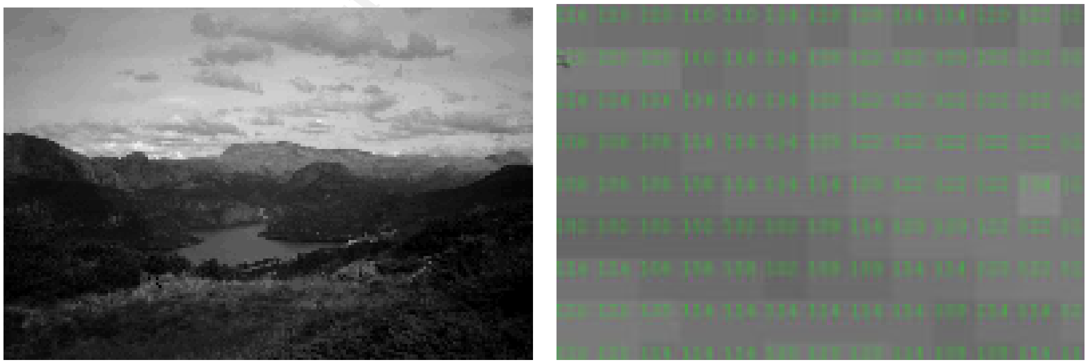


Figure 4-1: Cover Image and Uniform Pixel Area

As per instructions, I chose an odd number gray value that was close to the pixel values I was going to manipulate. I will refer to this value as "magic number" in this document. Using the calculator tool, I started to replace the pixel values in the image with the "magic number." I repeated this process until I was done typing the hidden message.

Figure 4-2 shows the image with a hidden message on the left hand side and the decoded message at the right hand side. The red circle on top of the mountain peak represent the area where the message is hidden. To decode the message, calculator tool is used again to convert bits lower than the "magic number" to "0" whereas, the higher ones are converted to "255."

This process converts rest of the image black and white, while revealing the hidden message in gray color. The quality of the hidden message used with pixel calculator is nowhere close to what S-Tools and Jsteg provide, but it helps in understanding some of the basics of image processing.

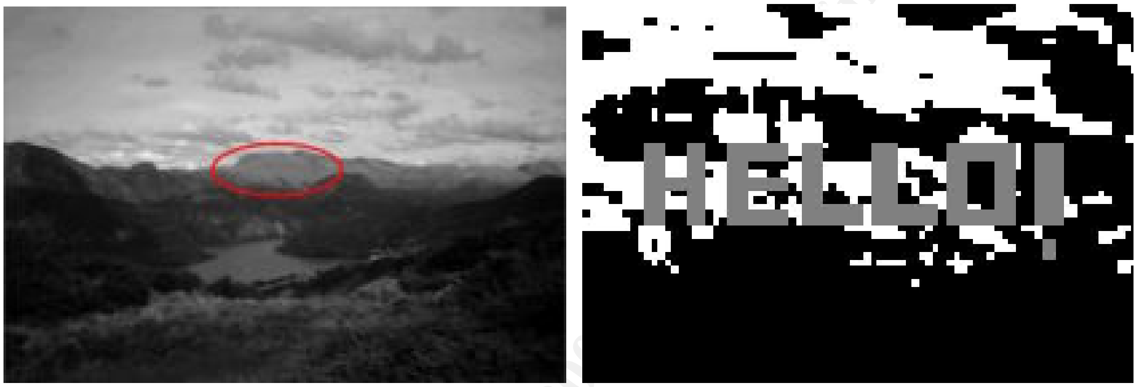


Figure 4-2: Encoded Picture and Decoded Message

S-Tools and Jsteg use different techniques than those used by the pixel calculator. S-Tools hides information in lossless images like Bitmap (BMP) and Graphic Interchange Format (GIF). S-Tools employs the LSB insertion method to hide the information within an image. Changes introduce by LSB insertion are imperceptible to human eye. For example if three 24-bit pixels are represented as below:

(00100111 11101001 11001000)
(00100111 11001000 11101001)
(11001000 00100111 11101001)

Than interesting letter "A", which is represented by (10000011) in binary, will result in the following: [14]

(00100111 1110100*0* 11001000)
(0010011*0* 11001000 1110100*0*)
(11001000 00100111 11101001)

I used a program called FileRay, to compare the binary values of two image files that were operated on using S-Tools. Image78.gif was used as the cover medium to hide Image79.gif. The original images are not provided in this document to save the space. The resultant file was named hidden.gif. Comparison of the original image to the stego data reveals changes in the LSB. Figure 4-3 shows the comparison between the two image files. The original image is shown in the bottom pane and the stego image is reflected in the top pane.

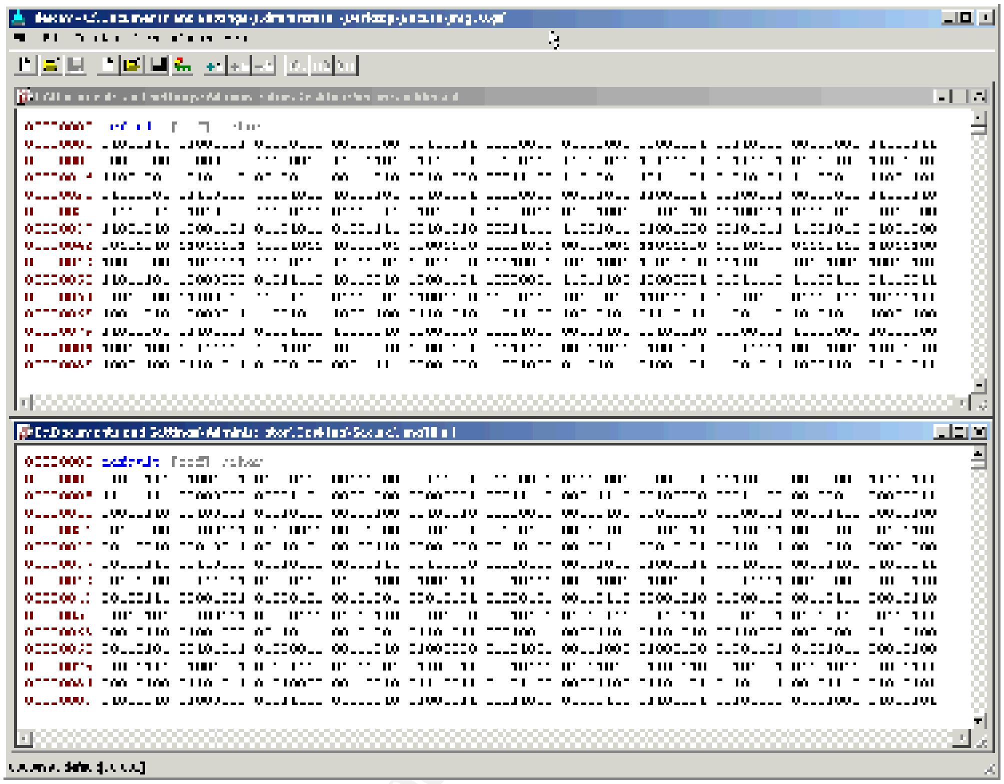


Figure 4-3: Comparison of cover image with stego in binary

Jsteg hides information in lossy image files and produces an output file with a Jpge extension. To accomplish this Jsteg employs algorithms and transformation techniques to change the Discrete Cosine Transform (DCT) coefficient in digital images. I spent more time in researching Jsteg and understanding Joint Photographic Expert Group (JPEG) images than other tools and graphic formats. More on Jsteg and Jpeg is presented in the Steganalysis section of this paper.

5 Steganalysis

As the techniques to hide information get more complicated and computationally involved, the detection of such cover medium has become considerably more challenging as well. However, given time, dedication and technology it is possible to detect the presence of hidden information in some stego mediums. A few tools have known signatures that may predict the presence of hidden information. Techniques like encryption and compression are used to make it difficult to decipher the hidden information. However, knowing the fact that there is hidden information present in the cover destroys the purpose of steganography.

Different mechanisms are used to detect the hidden information. In this paper I will explore a variety of such mechanisms. Some of these techniques are specific to a given tool, while others are more robust and are concerned with the statistics of medium versus the tool that were used to hide the information.

For example research work done at George Mason University (GMU) shows that some well-known tools like S-Tools have known signatures and can be recognized if proper techniques are used. S-Tools works by reducing the number of colors of the cover image to 32, but expands them over several color palette entries, if the palette is then sorted by luminance, blocks of colors appear to be the same, but actually have a one-bit variance. This type of variance pattern is extremely rare in a natural image [12].

As mentioned earlier, instead of focusing on steganographic algorithm, patterns in normal images can be understood to determine the abnormalities in images that deviate those properties. One such method is to understand the first order or the higher-order statistics of natural images and then discover the alteration caused by the hidden message in these statistics.

In his work at Dartmouth College Computer Science department, Hany Farid has developed techniques to determine the presence of embedded messages into digital images by understanding the higher-order statistics of natural images. To achieve this goal he used the low-pass and high-pass filters to decompose the images to split the frequency space into multiple scales and orientation [15].

After decomposing the image he collected a set of statistics to understand the natural images. He then used programs like Jsteg, EzStego and OutGuess to compress and hide information into natural images. His results show 97.8% detection rate for a 256x256 image embedded using Jsteg. EzStego, OutGuess- and OutGuess+ provided a detection rate of 86.6%, 80.4% and 77.7% respectively. A small percent of false positives were also generated.

In their research work Andreas Westfeld and Andreas Pfitzmann contradict the idea that the least significant bits of luminance values in digital images are completely random and could therefore be replaced [1] They developed a java application to experiment visual attacks on images hidden using tools like EzStego, S-Tools and Steganos. In their paper Andreas and Andreas explain mechanism used by EzStego to embed the data in digital images. Later they used this knowledge to develop filters to remove all parts of image covering hidden message. Results presented in this paper are remarkable.

5.1 Independent Research

I have been working to develop a mechanism to determine hidden information in Jpeg images. This research is not done yet, but some of the findings are presented in this paper. To perform this research I first had to understand Jpeg images and the coding techniques behind them. As the topic of this document is not graphical coding schemes, I will present only a short description of how Jpeg images are coded. This information is necessary in understanding the information provided in the next section.

JPEG is a lossy image compression method, where during the compression some of the information that human eyes are not sensitive to is dropped. In contras to lossy compression method there exists a lossless compression method, which preserves most of the information present in the original image.

In general computer graphic cards use red, blue and green as the primary colors to generate the colorspace. These three colors form the axis of a Cartesian coordinate. The JPEG coding procedure however, divides an image into 8x8 blocks of pixels in the Luminance, Red Chrominance, and Blue Chrominance (YUV) colorspace. Luminance axis is one where RGB have equal values. The relation between RGB color space and YUV color space is depicted in Figure 5-1 [6].

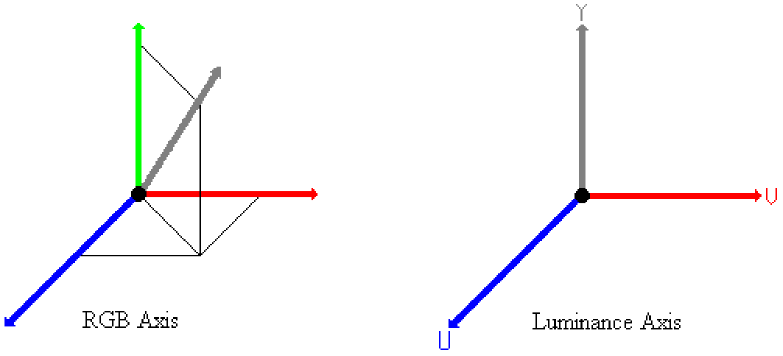


Figure 5-1: RGB Color Axis and Luminance Axis [6]

To translate between the two color spaces the following equations are used.

$$\begin{aligned}
Y &= 0.299\,R + 0.587\,G + 0.114\,B \\
U &= -0.1687\,R - 0.3313\,G + 0.5\,B + 128 \\
V &= 0.5R - 0.4187\,G - 0.0813\,B + 128
\end{aligned}$$

The Research has shown that Human Visual System (HSV) is more sensitive to luminance then chrominance. Therefore, when pixel blocks are run through a two dimensional discrete cosine transform (DCT) the resulting frequency coefficients are scaled to remove ¾ of the chrominance information. The DCT coefficients are calculated using the following relationship [8].

$$t(i,j) = c(i,j)\sum_{n=0}^{N-1}\sum_{m=0}^{N-1} f(m,n)\cos\frac{\pi(2m+1)i}{2N}\cos\frac{\pi(2n+1)j}{2N}$$

While JPEG is a lossy compression technique, one important thing to recognize is that the overall process is split into two stages. The lossy stages use a discrete cosine transform and a quantization step to compress the image data; the lossless stage then uses

Huffmann coding to further compress the image data. To reduce the noise and corruption risk steganography data is data is inserted between these two stages.

Prior to any encoding the DCT coefficients are unfolded into a one-dimension array using the zigzag pattern as shown in Figure 5-2 [8]. The low frequencies are ordered first in the one-dimensional array following the high frequencies most of which are set to zero to improve compression.

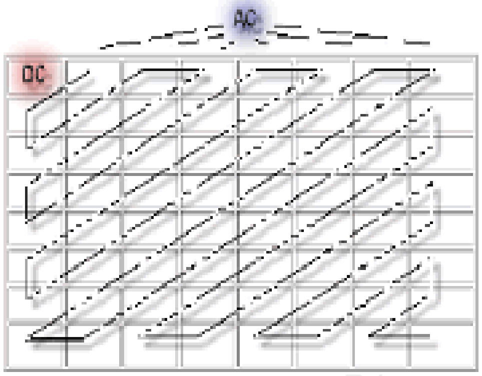


Figure 5-2: Zigzag Pattern [8[

Once the steganography data is loaded into the JPEG image, the lowest-order bits of all non-zero frequency coefficients are replaced with successive bits from the steganographic source file, and these modified coefficients are sent to the Huffmann coder [7].
JPEG compression process is depicted in Figure 5-3 [5]

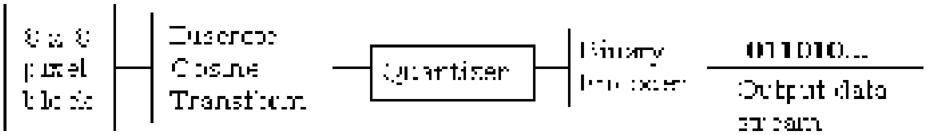
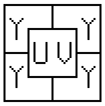


Figure 5-3: Block Diagram of JPEG image compression [5]

I used the Jsteg program to hide information to produce an output files in JPEG format. After acquiring basic understanding about JPEG images, I used a proprietary program to fetch the DCT coefficients out of the stego image file. As mentioned earlier these coefficient values are the ones modified to hide information within JPEG images. This process provided me with an Nx1 matrix of DCT coefficients. I collected a 64x64 sample (4096 coefficients) of this matrix to perform some preliminary research.

Next I used the Matlab program, by mathworks, to order the Nx1 matrix (4096 coefficients) into an array of 64x64. Again using the Matlab program I converted these coefficients in a zigzag pattern as mentioned earlier in this document. The luminance and chrominance coefficients were distinguished via the following encoding scheme considering that luminance was sampled at full resolution.



According to this scheme, typically, but not always the horizontal and vertical sample factors for Luminance are 2 and horizontal and vertical sample factors for both of the chrominance components is 1[6].

Next step involved comparing the neighboring low order frequencies to determine a relationship among them. I have yet to write code to compare such frequencies, but performing some work using calculator showed a close relationship among neighboring frequencies. This could be an indication of data hidden within a JPEG image, as the natural JPEG images do not necessarily have low-order frequency values that bare a close relationship with their neighbors. This is a work in progress and updates and results will be posted on the Internet later on this year.

6 Conclusion

Steganography has a rich history backing its strength and robustness. However, the advancements in computer technology have impacted steganography as equally, if not less, as many aspects of our daily life. While having different purposes than cryptography, steganography is complemented by cryptography in most current software tools.

The tools available to perform steganography are getting very sophisticated and user friendly yet providing strong means to hide the data. Increasing knowledge in digital signal processing and image format compliments the knowledge of algorithms and techniques used by steganography tools in detecting hidden information.

Software tools, such as Stegdetect, are easy to use tools in performing steganalysis, but are known to generate many false positive [9]. Many institutes and individuals are working towards improving steganography and steganalysis techniques and one if fair in saying that sky is the limit for Steganography.

References

1. Andreas Westfeld, Andreas Pfitzmann: Attacks on Steganographic Systems. Information Hiding, 3rd International Workshop, Proceedings, Dresden September/October 1999, LNCS 1768, Springer Verlag, Berlin 2000, 61-76

2. Twin Peaks: The Histogram Attack to Fixed Depth Image Watermarks. Information Hiding, 2nd International Workshop, IH'98 Portland, Oregon, USA, April 1998, pp 291-305

Internet Sources

3. The Steganographic File System: http://www-users.rwth-aachen.de/peter.schneider-kamp/sources/sfs/
4. The Pixel Calculator website: http://www.cs.washington.edu/research/metip/software/pixel.html
5. Welcome to JPEG Tutorial: http://www.ece.purdue.edu/~ace/jpeg-tut/jpegtut1.html
6. JPEG Compression page by Elmo: http://web.usxchange.net/elmo/jpeg.htm
7. Steganography archive: http://www.theargon.com/archives/steganography/DOS/jsteg%2Etxt
8. Jpeg basics at http://www.rasip.fer.hr/research/compress/algorithms/adv/jpeg/
9. Niels Provos, Peter Honeyman: Detecting steganography content on the internet http://www.citi.umich.edu/techreports/reports/citi-tr-01-11.pdf
10. Duncan Sellars: An Introduction to Steganography http://www.cs.uct.ac.za/courses/CS400W/NIS/papers99/dsellars/stego.html
11. Neil F. Johnson: In Search of the Right Image: Recognition and Tracking of Images in Image Databases, Collections, and the Internet http://www.jjtc.com/pub/csis_tr_99_05_nfj/
12. Neil F. Johnson and Sushil Jajodia: Steganalysis of Images Created Using Current Steganography Software
13. http://www.jjtc.com/ihws98/jjgmu.html
14. Neil F. Johnson, Sushil Jajodia: Exploring Steganography: Seeing the Unseen http://www.jjtc.com/pub/r2026.pdf
15. Hany Farid: Detecting Steganography Messages in Digital Images http://www.cs.dartmouth.edu/~farid/publications/tr01.pdf
16. Lecture Notes: Neil F. Johnson: Presented in INFS 762 at GMU
17. W. Bender, D. Gruhl, N. Morimoto, A.Lu: Techniques for Data Hiding http://www.research.ibm.com/journal/sj/mit/sectiona/bender.html