Wave Steganography

GRADUATE PROJECT TECHNICAL REPORT

Submitted to the Faculty of
the Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Sheshidhar Odeti

Fall 2007

Committee Members

Dr. Dulal Kar
Committee Chairperson

Dr. Mario A. Garcia
Committee Member

Dr. David Thomas
Committee Member
ABSTRACT

Steganography is the art of concealed communication. Steganography is derived from the Greek word “steganos”, which means hidden or secret and “graphy” means writing or drawing. Thus Steganography means secret writing. Steganography can be implemented in text, images, audio or video. The concealment of information is just not enough in some cases. The data should be encrypted and then hidden. Thus even if some one is able to recover the secret data from carrier file, the actual content is not revealed.

This technical report discusses a Stego tool which implements ‘Audio Steganography’ or ‘Wav Stego’ using a .wav file as carrier, the tool can embed any file in it, without altering the quality of the audio file used. The properties of the cover file still remains unchanged even after embedding any file into it.

This tool can be extended to other carrier file formats such as mp3, wma, wmv etc., and also the size of secret file to be hidden can be increased.
# TABLE OF CONTENTS

Abstract ................................................................................................................……….ii

Table of Contents ..................................................................................................……. iii

List of Figures .........................................................................................................……vi

1. Background and Rationale .................................................................................1
  1.1 Steganography .................................................................................................1
    1.1.1 Terminology ..............................................................................................1
    1.1.2 Variations in Stego ....................................................................................1
  1.2 History of Steganography ..............................................................................3
  1.3 Classification of Stego ....................................................................................4
    1.3.1 Insertion Based ..........................................................................................4
    1.3.2 Substitution Based ....................................................................................5
    1.3.3 Generation Based ....................................................................................5
  1.4 Steganography in Various Media .................................................................7
    1.4.1 Steganography in Text ..............................................................................7
    1.4.2 Steganography in Images .........................................................................11
  1.5 Least Significant Bit Algorithm .................................................................13
    1.5.1 24 Bit Images ...........................................................................................14
    1.5.2 8 Bit Images ............................................................................................15
    1.5.3 Masking and Filtering .............................................................................16
  1.6 Legitimate Use of Steganography .................................................................16

2. Narrative ............................................................................................................17
  2.1 Generic Process of Encoding and Decoding ..............................................17
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.5.2 Un-Hiding Secret Message</td>
<td>42</td>
</tr>
<tr>
<td>4. Evaluation and Results</td>
<td>44</td>
</tr>
<tr>
<td>4.1 White-box Testing</td>
<td>44</td>
</tr>
<tr>
<td>4.1.1 Unit Testing</td>
<td>44</td>
</tr>
<tr>
<td>4.1.2 Integration Testing</td>
<td>45</td>
</tr>
<tr>
<td>4.2 Black-box Testing</td>
<td>45</td>
</tr>
<tr>
<td>4.3 Validation Testing</td>
<td>46</td>
</tr>
<tr>
<td>4.4 Alpha and Beta Testing</td>
<td>49</td>
</tr>
<tr>
<td>5. Conclusion and Future Work</td>
<td>50</td>
</tr>
<tr>
<td>5.1 Usage of Encryption Algorithms</td>
<td>50</td>
</tr>
<tr>
<td>5.2 Usage of Carrier Files</td>
<td>50</td>
</tr>
<tr>
<td>5.3 Usage of Secret Files</td>
<td>51</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>52</td>
</tr>
<tr>
<td>Bibliography and References</td>
<td>53</td>
</tr>
</tbody>
</table>
## LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1</td>
<td>Example of Fractal</td>
<td>6</td>
</tr>
<tr>
<td>Figure 1.2</td>
<td>Binary Representation of ‘A’</td>
<td>14</td>
</tr>
<tr>
<td>Figure 1.3</td>
<td>Least Significant Bit Insertion</td>
<td>15</td>
</tr>
<tr>
<td>Figure 2.1</td>
<td>Generic Process of encoding and decoding</td>
<td>17</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Interfaces for Encryption and Decryption</td>
<td>19</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Selecting Secret File</td>
<td>20</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Selecting Carrier File (Wave File)</td>
<td>22</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>Selecting Output Filename After Hiding</td>
<td>23</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>Entering the Password to Generate Key</td>
<td>24</td>
</tr>
<tr>
<td>Figure 2.7</td>
<td>Hiding Process successfully Completed</td>
<td>25</td>
</tr>
<tr>
<td>Figure 2.8</td>
<td>Selecting Stego file (Wave file which has Secret File hidden)</td>
<td>26</td>
</tr>
<tr>
<td>Figure 2.9</td>
<td>Selecting Output file Name for Secret Message to be Retrieved</td>
<td>27</td>
</tr>
<tr>
<td>Figure 2.10</td>
<td>Un-hiding the Secret Message from Stego File</td>
<td>28</td>
</tr>
<tr>
<td>Figure 2.11</td>
<td>Un-hiding Process Successfully Completed</td>
<td>29</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Wave File Format</td>
<td>31</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>Process of Hiding Secret File into Audio File</td>
<td>33</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>Process of Retrieving Secret File from a Wave Stego</td>
<td>33</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Sample Wave File</td>
<td>37</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>E Sample Binary Bits</td>
<td>37</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Sample Audio Bits Showing Disposition Bits</td>
<td>39</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>Audio Bits after Substituting Secret File Bits</td>
<td>39</td>
</tr>
<tr>
<td>Figure 3.8</td>
<td>First 32 Data Bytes of Audio File</td>
<td>41</td>
</tr>
</tbody>
</table>
Figure 3.9  First 32 Data Bytes of Audio file with Secret File Size…………………42
Figure 4.1  Initial Size of Wave file/Carrier file………………………………………46
Figure 4.2  Initial Size of Image before Hidden into Audio/Wave file………………47
Figure 4.3  Initial Image Content…………………………………………………….47
Figure 4.4  Wave File Size after Hiding Image………………………………………47
Figure 4.5  Image File Size after Retrieving from Wave…………………………….48
Figure 4.6  Image Content after Retrieving from Wave file…………………………48
1. BACKGROUND AND RATIONALE

1.1 Steganography

1.1.1 Terminology

According to the Information hiding workshop held in University of Cambridge, England, “cover”, “embedded” and “stego” are the three adjectives used to explain the definition of steganography. The term “cover” is used to describe the original, innocent message, data, audio, still, video and so on. The information to be hidden in the cover data is known as the “embedded” data. The cover signal and embedded information together is referred to as “stego”.

1.1.2 Variations in Stego

The motive of pure steganography is to communicate data secretly with somebody. However, there are subsets of the steganography where the motive may vary but still the data is in hidden state. Trojan horses and Easter eggs are two variations which are subsets of steganography. The recipient may not be happy after uncovering the hidden data in these cases [Brainos 2006].

**Trojan Horses**

Trojan Horses are the malicious programs often used by attackers to creep viruses into an organization. This program has an overt and a covert part. The user can view only an overt part which is mostly an animation game. The user is unaware of the covert part. The covert is the program that is installed in the back ground of an overt program. This program is a malicious program or a virus through which the attacker wants to gain the control of a system which executes it.
The similarity between a Trojan horse program and that of steganography is that there is an overt and a covert program in the steganography where the true intent of the program is hidden from the user.

There are differences in Trojan horse program and pure steganography. In pure steganography the receiver automatically receives the message. In the case of Trojan horses the data has to be embedded manually and the user also execute the program i.e; he or she has to extract the hidden message manually. Trojan horses put the pay-load or the malicious program in the overt program. The pay load is passive. Thus, Trojan horses are specialized subsets of stego as they always keep the pay-load in executable files only and will actively do something against the destination host [Cvejic 2004].

The other difference is that in case of steganography the sender and recipient are always aware of the intent of the communication. In the case of Trojan horses only the attacker has the complete information about the covert and overt program. The recipient is unaware of the intent of the program.

*Easter Eggs*

Easter eggs are a hybrid between Trojan Horses and stego. An Easter egg is a hidden feature such as animation or video that the developer of an operating system or application sneak into the program. At some point in time the programmers reveal the existence of the Easter egg to the users. Easter eggs are usually just for fun.

Easter eggs operate very similarly to stego. In stego and Easter eggs someone inserts a covert program into an overt, and someone has to follow a specific set of steps to remove or extract it. These steps, usually involve opening up an application, going to a certain area of it, and typing specific words or commands. Other common Easter eggs
involve making a modification to an application’s configuration file and then starting up the application. The intent is not malicious, like that most of Trojan Horses, and these program do not automatically run without the user’s consent. [Cole 2003]

1.2 History of Steganography

A [Cox, Miller, Bloom 2001] example of steganography is a story from Herodotus, who tells of a slave sent by his master, Histiaeus, to the Ionian city if Miletus with a secret message tattooed on his scalp. After tattooing, the slave grew his hair back in order to conceal the message. He then journeyed to Miletus and, upon arriving, shaved his head to reveal the message to the city’s regent, Aristagoras. The message encouraged Aristagoras to start revolt against Persian king. In this scenario, the message is of primary value to Histiaeus and the slave is simply the carrier of the message.

In 1499 Trithemius published Steganographia, one of the first books about steganography [Katzenbeisser, Petitcoals, 2000]. Techniques such as writing between the lines of a document with invisible ink created from juice or milk, which show only when heated, were used as far back as ancient Rome. In World war II, Germany used microdots to hide large amounts of data on printed documents, masquerading as dots of punctuation.

In World War II, the following message was sent by a German spy: “Apparently neutral’s protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue effects pretext for embargo on by products, ejecting suets and vegetable oils”.

The message was interpreted as

“Apparently neutral’s protest is thoroughly discounted and ignored. Isman hard hit. Blockade issue effects pretext for embargo on by products, ejecting suets and vegetable oils”.

3
Taking second letter in each word the following message emerges:

“Pershing sails from NY June 1”. [Armstrong, Yetsko, 2004]

1.3 Classification of Stego

Steganography is classified into three categories based on “how to” and “where to” hide the secret data into cover file. They are:

- Insertion-based
- Substitution-based
- Generation-based

1.3.1 Insertion-Based

“Insertion based techniques find places in a file that are ignored by the application that reads the file”[Cole, 2003]. By using this technique one can insert the data into a text, audio, or video file without changing it. This can be achieved by hiding the data which is not being used by the file. For every file there is a end-of-file (EOF) marker. The data that is entered into a file after EOF will still be in the file but not displayed with the file while reading.

For example, let us consider two word documents. Let us take a empty document and type “Hello World”. Now let us take another document which is big. Let us assume it has 80 pages. Now let us type “Hello World” at the end of the document. Then let us delete all the contents before this text. Now the two documents look alike. The difference lies in the size of the documents. The second document which is modified is larger than the first.

Word documents have end of text and begin of text markers. In general the scanner of a word document will scan it between the start marker and the end marker, and
the content between the end text marker and begin text marker will be ignored. Thus this method doesn’t change the host file, but large amount of data can be inserted which is an advantage. But by having a word document of size 4 MB and which appears to contain just a single sentence like “this is a car” one can notice the possibility of a stego.

1.3.2 Substitution-Based

The substitution based scheme substitutes already present information with the data to be hidden. The substitution is to be done carefully so that it doesn’t change the actual file. The technique is to identify the unimportant/insignificant matter in the file and replace it with the data to hide.

For example, the information present in between the start and end text marker will be scanned. The information in between the end text marker and the start text marker is not scanned and is unimportant information. This information can be substituted with the data to be hidden.

The advantage of this method is that the size of the file is not changed. The disadvantage is that only a limited amount of data that can be hidden.

1.3.3 Generation-Based

Substitution based and generation based techniques require that the both host file and the data file be hidden. Host file is called the overt file and the hidden data is the covert file. Using Generation based stego one create an overt using covert file.

The problem such as: if a person has both files, namely one the original file and the other with data hidden in it, he or she can compare them and notice the stego in it. This problem is addressed by generation based stego methods.
The most common example of a generation based stego is to use an overt file to create a fractal image. “A fractal image has critical mathematical properties, but it is essentially a collection of patterns and lines in different colors” [Cole, 2003]. Figure 1.1 shows an example of a fractal image. Covert message is used to determine the angle length and color of each line.

To avoid detection in generation-based stego, the image generated has to make some sense and should be related to the person generated this, in some way. The generated stego file must have a predictable pattern that could be coded into an algorithm. For example, with a fractal image, the angles and length of lines can be fed into an algorithm that is based on the input text to create the covert image [Cole, 2003].

The main media for generation based stego in most cases is random looking images like fractals and English text. Some realistic pictures can’t be generated using generation based stego.

![Fig 1.1 Example of Fractal Taken from http://members.lycos.fr/warey/fract/str08.htm](http://members.lycos.fr/warey/fract/str08.htm)
1.4 Steganography in Various Media

1.4.1 Steganography in Text

One problem identified by Brassil and others is the illegal distribution of documents through modern electronic means, such as electronic mail. Means such as this allow infringers to make identical copies of documents without paying royalties or revenues to the original author. To counteract this possible wide-scale piracy, Brassil et al. [Brassil, 1994] discusses a method of marking printable documents with a unique codeword that is indiscernible to readers, but can be used to identify the intended recipient of a document just by examination of a recovered document.

The techniques they propose are intended to be used in conjunction with standard security measures. For example, documents should still be encrypted prior to transmission across a network. Primarily, their techniques are intended for the use after a document has been decrypted, once it is readable to all. An added advantage of their system is that it is not prone to distortion by methods such as photocopying, and can thus be used to trace paper copies back to their source.

An additional application of text steganography as suggested by Bender, et al is “annotation”, [Bender,Gruhl 1996] which can be checked to verify the document whether it has been tampered or not. An annotation is a comment inserted by a programmer in a source code. If this annotation is hidden using steganography, the corrupted files can be detected. For example, if a programmer has written a source code with hidden annotation and sent it over the Internet, the receiver knows that the file is corrupted by decrypting the annotation if this file is corrupted by an attacker in transit.
Hidden data in text could be used even by mail servers to check whether documents should be posted or not. The marking techniques as [Brassil, 1994] described can be applied either to an image representation of a document or to a document format file, such as PostScript or TEX files. The idea is that a codeword (such as binary number, for example) is embedded in the document by alerting particular textual features. By applying each bit of the codeword to a particular document feature, one can encode the codeword. It is the type of feature that identifies a particular encoding method. Brassil identifies three features that are described in the following subsections:

1) Line-Shift Coding
2) Word-Shift Coding
3) Feature Coding

**Line-Shift Coding**

In this method, text lines are vertically shifted to encode the document uniquely. Encoding and decoding can generally be applied either to the file format of a document, or the bitmap of a page image. By moving every second line of document either 1/300 of an inch up or down, Brassil et al. [Brassil, 1994] found that line-shift coding worked particularly well, and documents could still be completely decoded, even after the tenth photocopy. However, this method is probably the most visible text coding technique to the reader. Also, line-shift encoding can be defeated by manual or automatic measurement of the number of pixels between text baselines. Random or uniform re-spacing of the lines can damage any attempts to decode the codeword. However, if a document is marked with line-shift coding, it is particularly difficult to remove the encoding if the document is in paper format. Each page will need to be rescanned,
altered, and reprinted. This is complicated even further if the printed document is a photocopy, as it will then suffer from effects such as blurring, and salt and pepper noise.

**Word-Shift Coding**

In word-shift coding, code words are coded into a document by shifting the horizontal locations of words within text lines, while maintaining a natural spacing appearance. This encoding can also be applied to either the format file or the page image bitmap. The method, of course, is only applicable to documents with variable spacing between adjacent words, such as in documents that have been text-justified. As a result of this variable spacing, it is necessary to have the original image, or to at least know the spacing between words in the un-encoded document. The following is a simple example of how word-shifting might work. For each text-line, the largest and smallest spaces between words are found. To code a line, the largest spacing is reduced by a certain amount, and the smallest is extended by the same amount. This maintains the line length, and produces little visible change to the text. Word-shift coding should be less visible to the reader than line-shift coding, since the spacing between adjacent words on a line is often shifted to support text justification. However, word-shifting can also be detected and defeated, in either of two ways.

If one knows the algorithm used by the formatter for text justification, actual spaces between words could be then be measured and compared to the formatter's expected spacing. The differences in spacing would reveal encoded data. A second method is to take two or more distinctly encoded, uncorrupted documents and perform page by page pixel-wise difference operations on the page images. One could then quickly pick up word shifts and the size of the word displacement. By re-spacing the shifted words back
to the original spacing produced under the formatter, or merely applying random
horizontal shifts to all words in the document not found at column edges, an attacker
could eliminate the encoding. However, it is felt that these methods would be time-
consuming and pain-staking.

Feature Coding

A third method of coding data into text suggested by Brassil et al. [Brassil, 1994] is
known as feature coding. This is applied either to the bitmap image of a document, or to a
format file. In feature coding, certain text features are altered, or not altered, depending
on the codeword. For example, one could encode bits into text by extending or shortening
the upward, vertical endlines of letters such as b, d, h, etc. Generally, before encoding,
feature randomization takes place. That is, character endline lengths would be randomly
lengthened or shortened, then altered again to encode the specific data. This removes the
possibility of visual decoding, as the original endline lengths would not be known. Of
course, to decode, one requires the original image, or at least a specification of the change
in pixels at a feature. Due to the frequently high number of features in documents that can
be altered, feature coding supports a high amount of data encoding. Also, feature
encoding is largely indiscernible to the reader. Finally, feature encoding can be applied
directly to image files, which leaves out the need for a format file. When trying to attack
a feature-coded document, it is interesting that a purely random adjustment of end line
lengths is not a particularly strong attack on this coding method. Feature coding can be
defeated by adjusting each end line length to a fixed value. This can be done manually,
but would be painstaking. Although this process can be automated, it can be made more
challenging by varying the particular feature to be encoded. To even further complicate
the issue, word shifting might be used in conjunction with feature coding, for example. Efforts such as this can place enough impediments in the attacker's way to make his or her job difficult and time-consuming.

### 1.4.2 Steganography in Images

**Color Tables**

Color tables are used to hide data in images. Several stego techniques use the color tables. All the images are composed of pixels. Each pixel is a dot, and combining all pixels form an image. Red, Green, and Blue are mixed in various percentages to get pixels a color. The possible variation of Red, Blue, and Green is from 0 to 255, where 0 indicates presence of a color and 255 indicates the saturation of colors. The combination of Red, Green, and Blue is represented as RGB. The total possible colors are obtained by combining these three colors in different concentration. Thus there exist \(256 \times 256 \times 256 = 16,777,216\) different colors.

Values of some common colors:

<table>
<thead>
<tr>
<th>Red</th>
<th>Green</th>
<th>Blue</th>
<th>Output color</th>
</tr>
</thead>
<tbody>
<tr>
<td>255</td>
<td>0</td>
<td>0</td>
<td>Red</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Black</td>
</tr>
<tr>
<td>255</td>
<td>255</td>
<td>255</td>
<td>White</td>
</tr>
</tbody>
</table>

The RGB value for each color is stored in a color table. Each entry includes a row number, a value for red, green, and blue [Cole, 2003].

**Image Representation Types**

An image can be a 8-bit GIF or a 24 bit bit-map. If the image is 8 bit, then it has \(2^8\) colors and 24 bit image has \(2^{24}\) colors in it. Larger images suit well for steganography, as
the amount of data one can hide increases with the size of image. Along with larger size 24 bit map images are highly recommended for stego techniques. There are some problems with larger images. It becomes a burdensome task to download and upload images and also drags attention of attackers due to bulk size transfer which is usually uncommon.

Some techniques can be used to reduce the size of the image. The most typical techniques to compress the size of bit map image

- GIF
- JPEG

**GIF**- “The GIF (Graphics Interchange Format) is a bitmap image format for pictures with up to 256 distinct colors from the over 16 million representable in 24 bit rgb. GIF s are compressed files, which reduces the amount of time it takes to transfer images over a network connection” [Wikipedia, 2006]. The 24 bit file format is reduced to an 8 bit format by using GIF. GIF employs lossless data compression, which ensures if an image fits 256 color formats then it can be compressed without any loss of visual quality.

**JPEG**- “JPEG is a commonly used standard method of lossy compression for photographic images. The file format which employs this compression is also called as JPEG; the most common file extensions for this format are .jpeg, .jfif, .jpg, .JPG or .JPE although .jpg is the most common on all platforms”. [Wikipedia, 2006]

Though some times JPEG may corrupt the hidden data, it does a great deal of job in reducing the size of the image and also in retaining the quality of the image. “The
JPEG 8 pixel by 8 pixel blocks and performs a 64 bit DCT (Discrete Cosine Transformation) which does not compute to exact values” [Francesco Queirolo, 2001].

**Palette and Image Composition**

It is an important task to select the right combination of palette and image composition. The image with grey scale colors best suits for stego techniques rather than an image with solid colors. This is because as one hides data in a grey scale, the pixels gradually change the color and thus one can introduce little error. If one uses an image with image with solid colors, and the image changes drastically with a little amount of error introduced.

**Three Basic Techniques to Hide Data in Image**

There are many ways to hide a message in the image. The message can be hidden in the noisy areas of the image or in those areas where there is a great deal of color variation. By doing this, attacker attention may be minimized. The message can be scattered all over the image and inserted directly. [Sellars, 2004]

There are three common approaches to hide a message in an image:

- Least significant bit insertion
- Masking and filtering, and
- Algorithms and transformations

**1.5 Least Significant Bit Insertion**

This is one of the most common approaches used in hiding the messages in the image. This method inserts the covert information directly into the cover file. The problem with this method is-it is vulnerable to even slight manipulation of the image. The conversion of JPEG to GIF is lossless compression method, i.e., there is no change to
image or data hidden in it. But the conversion of GIF to JPEG is a lossy compression method and the image changes and also the message hidden is lost. Twenty four bit images and 8 bit images are used in least significant bit insertion process.

### 1.5.1 24-bit Images

Hiding a message in a 24 bit image, insert a bit in every byte of information. This is done by inserting the message to be hidden in every least significant bit. Thus when one takes a pixel, which is of size 3 bytes, it can store 3 bits of information in it. “A 1,024 * 768 image has a potential to hide a total of 2,359,296 bits of information” [Johnson 1998]. By compressing the message prior to hiding allows large amount of information to be hidden in a message. By hiding such a large message in an image does not manipulate the image and it looks still the original image to the human eye. For example, the letter A can be hidden in 3 pixels of an image. The binary value for A is 10000011 as shown in figure 1.2 below. The 3 pixel information may be

![Fig 1.2 Binary Representation of “A”](image)

Now insert A into LSB of the above three pixel image. The 3 pixels change as follows after embedding A as shown in figure 1.3 below.
Fig 1.3 Least Significant Bit Insertion

Now if one examines the image that changed after inserting A in it, the change is minute and only 4 bits have to be changed on the whole. On an average when one inserts a message, less than half of bits have to be changed, thus change is minor and the image does not change drastically. One can even store the data in the least significant bit and second least significant bit and the change can not be noticed by a human eye.

1.5.2 8-Bit Images

Many techniques proposed by authors to hide messages in 8 bit images. These techniques are drawn from the practical approaches and seem to be better as the new ways to hide were discovered. Hiding a message in an 8-bit image is difficult task when compared to 24-bit image. “Since 8 bit values can only have a maximum of 256 colors, the image must be chosen much more carefully” [Francesco Queirolo, 2001]. For example, take a palette with 4 colors: white, red, blue, and green which have the palette position entries of 0(00), 1(01), 2(10) and 3(11) respectively. If a decimal number 11 represented as 1011 is hidden in this palette of four colors, upon hiding 1011 in LSB of the palette, it changes as (01 01 11 11) with the adjacent colors becoming same and two colors completely vanishing. Thus the changes are drastic in an 8 bit image. The colors have to be chosen carefully and also gray scale palette offers more resistance to changes [Francesco Queirolo, 2001]. The implementation of LSB in an image can be done by using stego tools.
1.5.3 Masking and Filtering

This technique is similar to that of water marking. In watermarking one encodes some message on the cover which can be noticeable to everyone. “Digital watermarks may include such information as copyright, ownership, or license” [Johnson, 1998]). Digital watermarks in general embed some additional information to the original image, so that they can be visible by naked eye and provide the ownership symbol. This is where the steganography differs. Steganography is used to hide the message embedded, and the means of communication is only cover. The embedded message has to be hidden and shouldn’t be visible. For example, a watermarking technique such as embedding additional information in currency to make them visible to every one to detect original from duplicates. If one takes the image of the currency and somehow change the colors of the image the watermarks embedded on it will be no longer visible. This technique is called Masking and Filtering. “Masking is more robust than LSB insertion with respect to compression, cropping, and some image processing. Masking techniques embed information in significant areas so that hidden message is more integral to the cover image than just hiding it in the “noise” level” [Shien Lu 2005]. This makes masking to preferable to LSB in hiding message in images.

1.6 Legitimate Use of Steganography

Post attack on World Trade Centers, there has been a lot of speculation on use of steganography by terrorists. It is believed that most of the information, they communicated is through stego techniques. It is important that study should be done with basic ethics. [www.sans.org, 2002]
2. NARRATIVE

This project “Wave Steganography” implemented steganography in audio files. The tool designed known as ‘Audio Stego’ or ‘Wav Stego’ hides any file into a .Wav file (Audio file). The basic purpose of this tool is secret communication between two entities. This can be achieved by a successful implementation of Wave Steganography.

Steganography requires two media to implement a secret communication. Here the tool designed takes .Wav file and any file as two media. The .Wav file being the innocent file or cover file and any other file is the embedded file or actual message that needs to be communicated.

Below is the figure 2.1 which shows generic process of wave steganography.

2.1 Generic Process of Hiding and Un-hiding

Fig 2.1 Generic Process of encoding and decoding
2.1.1 Encrypt

The system takes audio file (.Wav) as the input in which data is hidden. It then accepts another file. The encrypt method encrypts the second file before hiding these encrypted bits in Wave file. For encrypting the file the system may use different techniques of encryption.

2.1.2 Key

During encrypting a secret message into a cover file one may use a secret key which may be private or public. If a key is public it will remain same for all encrypting and decrypting all files on a particular stego tool. If a key used is public, they separate keys can be used for different hidden files. This key will later help in decrypting the secret image from the cover file. In this project, the key is generated based on the password used by user before hiding. This password is only known to intended recipients and uses it during retrieval of hidden file.

2.1.3 Hide

Based on different criteria, the encrypted file bits are hidden into audio bits. This hiding process may follow different patterns. Here encrypted secret file size and audio file size are used as criteria to determine where to hide secret file bits.

2.1.4 Un-hide

One needs to unhide the bits during the process of retrieving the secret file from cover file. This follows reverse process of hiding, and based on same logic used during hiding of bits.

2.1.5 Decrypting
Once the hidden data is recovered from carrier file it needs to be decrypted, since it is in cipher form. Once the hidden bits are retrieved from stego file, then these bits are decrypted using key that has been created during encryption to make it a meaningful message/original secret file, from decrypted data.

2.2 Actual Process of Hiding Secret Message

The Wav Stego tool developed takes two input files .Wav file and any other file.

![Fig: 2.2 Interfaces for Encryption and Decryption]

Here the tool asks the user to input Image file and Carrier file as shown in figure 2.2.

2.2.1 Selecting Secret File
The tool can accept any secret file. The user selects the “Browse” option and can select any file from the system. Once the secret file is selected then user hits on open. Figure 2.3 depicts this.

Fig: 2.3 Selecting Secret File
2.2.2 Selecting Carrier File

Here the tool needs a carrier file to hide the any secret file into it. The carrier file in this case would be a .Wav file into which the tool embeds the secret file. The user can select a Carrier file in a similar way to secret file selection as shown in figure 2.4.
Fig: 2.4 Selecting Carrier File (Wave File)
2.2.3 Selecting Output File

The user has to select “Browse” as shown in figure 2.5, to select the output file destination to save stego file, i.e., the file which will have audio file and hidden secret file in it. The output file format will be .Wav which is default.

Fig: 2.5 Selecting Output Filename After Hiding
2.2.4 Hiding Secret File

User has to select a password before he or she can start hiding process. The password needs to be entered twice. The password field can’t be left blank. If the password is left blank, the tool prompts the user to enter a password. The user needs to select ‘Hide’ to start the encryption process as shown in figure 2.6. Once the encryption process completes a message pops up saying “Hiding Process Successfully Completed” as shown in figure 2.7 below.
2.3 Process of Un-hiding Secret Message

In the tool developed it has a select option where a user can select either “Hide Secret File” or “Secret File Retrieval”. The default is Hide Secret File. To start retrieval process, select the radio button “Secret File Retrieval”. Now that the hiding is completed one has to select the wave stego file that has secret file hidden in it.

2.3.1 Selecting Encrypted/Stego file

The user has to select the browse button and need to locate the saved encrypted file and open it. This file has the .Wav file format. This is shown in figure 2.8.
2.3.2 Selecting Output File

Now the original, secret file from the encrypted file needs to be extracted. So after selecting browse to select a destination and user has to specify the file name and extension as shown in figure 2.9.
2.3.3 Secret File Retrieval

To ensure the retrieval user has to enter the password twice and should match the original password used during hiding of secret file. To start retrieval of the secret file from audio file, user has to select unhide button as shown in figure 2.10. Once the un-
hiding process is completed the output file is saved into the destination folder selected under the selected name.

Fig: 2.10 Un-hiding the secret message from Stego File

Once the process of un-hiding is completed one can see the secret message being retrieved in the path selected for the secret file. This secret message will be in the name of selected secret file name. For the above example it will be in name of “newtest.jpg”.

Once the process is completed the tool displays “UnHiding Process Successfully Completed” message to indicate that the secret file is ready to access as shown in figure 2.11.
Fig 2.11 Un-hiding Process Successfully Completed
3. SYSTEM DESIGN

The tool designed can hide any file into an audio file. Here any file can be used as a secret file, and .Wav file as a carrier file. The system first takes an audio file as carrier file, and accepts any secret file to hide into it. It uses Triple DES encryption technique to encrypt secret file before hiding into audio. Triple DES generates a key based on the user password, and thus the password is external to system and not saved either in code or stego file generated. This can be given to recipient through a different medium of communication to ensure security. Thus the recipient needs the password, the original source code and the file extension of the hidden file to retrieve the secret message.

3.1 Code Design

The system is coded in Visual C++ platform. This is executed in Visual Studio 6.0 environment.

3.2 Wave File Format

Basically, wave file is made of number of different types of chunks. The wave file format is shown in figure 3.1. The important chunks being

1) RIFF Chunk
2) fmt Chunk
3) data Chunk

Of these chunks, any application reading a wave file must be able to read at least two chunks namely “fmt chunk” and “data chunk”.

3.2.1 RIFF Chunk

This chunk gives us the information about the type of file; here the type is “wave”. This chunk consists of a single wave containing 2 required chunks, a fmt chunk and data chunk.

3.2.2 Format Chunk

A format (fmt) chunk basically describes the properties of a wave such as sample rate, bit resolution, and number of digital audio stored in a wave.

3.2.3 Data Chunk

The data chunk contains actual sample frames or the actual data of the wave. All the data are stored in the form of 8-bit (bytes) format. The bits of multiple bytes of data are stored with the low-order (least significant) bytes first.

Fig 3.1 Wave File Format
3.4 Cryptography Algorithms to Use in Steganography

With the advent of Information age heralded a new era of free flow of information. Like in any other fields this free flow of information has its own good and bad, good being the easy accessibility of the information and bad being the lack of security to the Information. Hence during the early 70’s researchers started concentrating on security of data which resulted in the development of cryptographic algorithms. Cryptography itself originated from Two Greek Words “Kryptos” meaning the “hidden” and “Graphy” meaning the writing” commonly referred as the hidden writing. Hence the main goals of Cryptography can be referred as Data Integrity, Data Origin Authentication, Data Confidentiality, Non Repudiation of the Origin and User Authentication.

In Cryptography, one uses encryption where in the plain text is converted into a cipher text using a set of rules or algorithms. Then in order to derive the original text, simply use the inverse of the algorithm or procedure for deriving the plain text from the cipher text. This process need a secret parameter called as key which is used for the encryption and decryption processes and is known only to the sender and the receiver. Hence it is very important to manage one’s keys.

There have been a lot of algorithms developed by various organizations and individuals but the some of most frequently used are:

1) AES (Advanced Encryption Standard)
2) TDES/TDEA (Triple Data Encryption Standard/Algorithm)
3) DES (Data Encryption Standard)
TDES algorithm is used here to encrypt the secret file before embedding into audio file to ensure more security. The inverse technique of TDES is used for proper retrieval of the secret file.

3.5 Wave Steganography

Figure 3.2 and figure 3.3 combined depict the process of steganography.
The process of wave steganography is hiding some data into wave files. The purpose of this project is to hide any kind of file into an audio file, and transmit the audio file over the network and prevent others from knowing the secret data transmission. Thus, any person on the network may interpret an audio file being transmitted, but the actual message hidden in audio file is invisible.

Any file is only interpreted as 0’s and 1’s by any computer. Each 0 or 1 is called a bit and 8 such bits form a byte. The data stored in computer is in bytes. This is the basic concept used for developing this tool. This tool first accepts a .wav file and converts it into bits. A wave file is stored in little-endian format, i.e., the LSB (Least significant bit) appears first and ends with MSB (Most significant bit).

For example consider the following 2 bytes of data:

```
0 0 0 1 0 0 0 0
lsb msb
0 0 1 1 1 1 0 1
lsb msb
```

The tool accepts any file as a secret file to be hidden into audio file. The first step is to convert secret file into bits. The tool prompts the user to enter a password. This password is used to generate a key. The tool makes use of TDES to encrypt the secret file data. “The simplest variant of TDES operates as follows: DES(k3;DES(k2;DES(k1;M))), where M is the message block to be encrypted and k1, k2, and k3 are DES keys”.

[Wikipedia, 2007]

Based on the password entered by the user, the tool generates a key. This key is used by TDES to encrypt the data. Consider M is the secret file data in bits, and K is the key generated based on user password.

TDES calls DES function using parameters K, M. DES returns encrypted message block M1. Since the tool designed makes use of one external password, only one key is
generated. Thus TDES makes a second function call DES (M1, K), using the same key and generates the encrypted message M2. TDES again makes a function call DES (M2, K) and finally generates the encrypted message M3 from the original message M.

[www.troposoft.com, 2007]

The tool calculates the size of M in bytes, and stores the size in the first 32 data bytes of the audio file. This information can only be stored in 32 lsb bits available. For example if an encrypted file has a size of 1024 bits, which is equivalent to “10000000000” in binary bit representation. This is padded with 0’s on left to make it a 32 bit number. 10000000000 after padding becomes as “00000000000000000000010000000000”. This is stored in 32 least significant bits available. As 32 bytes of data is used to store size of encrypted secret file, some space is lost to store the actual encrypted secret file into audio. The system uses LSB substitution technique to hide secret file in audio.

Now the actual space available in audio to hide secret data is calculated as:


This gives data space available in audio file to store secret data. Since the tool makes use of LSB substitution it can hide a secret file which is up to 1/8 of this space available. If the user wants to hide a secret file which is larger than 1/8 of the available space, then user get an error saying, “secret file too big to hide”.

The substitution of secret data into audio bits is based on the ratio of audio space available to encrypted secret file size. The hiding technique is explained in detail in coming sections. The simplicity of this algorithm is that the tool do not disturb the audio
file header. Thus the properties of audio file remain same and can’t be distinguished from original audio file, by any person.

Once the secret file is encrypted into audio file, this can be transmitted over the network to the known recipient who has the same tool. To make this tool more secure, it is preferred not to send encrypted file and key together. It would be better if user can transmit the key through a different medium of communication to recipient, so that there is more security.

During the retrieval process, the user needs to select stego wave file and also specify the extension for the secret file (as the tool can encrypt any file into audio file). The tool prompts the user to enter a password, and the user has to provide the same password used during hiding process.

1) Hiding Secret Message
2) Retrieving Secret Message

3.5.1 Hiding Secret Message

This tool can hide any type of file into a wave file. Hiding secret file into wave file is accomplished in following steps:

1) Converting Wav file and secret file into binary Data.
2) Using some cryptic Algorithm to encrypt the secret data.
3) Arranging the encrypted data into Audio in some pattern.
4) Generate a stego output file as a wave.

**Converting Wav file and Image file into binary bits**

A function has been written to convert the wave file and the secret file into binary bits. This information is needed to calculate the size of the audio and secret files.
combined. The secret file size and Wav size are to be remembered for further use.

Sample wave file is shown in figure 3.4 and sample binary bits are shown in figure 3.5.

Cryptic Algorithm to Encrypt Secret File Data
Here the algorithm used is Triple Data Encryption Standard algorithm where in the data D in 64 bit format is accepted and a corresponding 64 bit cipher block C is generated using the encryption mechanism and a key. The Key is generated corresponding to the password entered by the user during encryption. For every 64 bit of data, this algorithm takes as 16 data blocks of hexa-decimal values and encrypts in 16 rounds and generates a cipher block of 64 bit.

This cipher block is now ready to replace or place in the wave file. Decryption then follows the reverse order from cipher block to the data block.

**Arranging the Encrypted Data into Audio**

Once the secret file data has been encrypted, it now needs to be hidden in audio bits. This is done by first calculation wave file size and encrypted secret file size. The code designed first calculates Disposition which is equal to “(Wave file Space Available in bits)/ (Secret file size in bits)”. This ratio should be less than “0.125”. Or mathematically the wave file bits should be at least 8 times or more than the image bits. In case if encrypted secret file is larger than 1/8 of the wave file in size, then the code gives an exception that file to be hidden is too large.

Once the disposition is calculated, this number is used for substituting pattern of audio bits with secret file bits. For example, let wave file size is greater 20Mb and secret file size is 2Mb. Let the space available in wave file be equal to 20Mb after removing 32 data bytes (used for saving information about encrypted secret file size), and header of wave. Now the disposition is equal to 20/2 = 10. So it moves forward by this disposition. The tool only replaces the LSB of the byte encountered by moving forward using disposition.
Let us consider the following is the bit pattern of an audio file as shown in figure 3.6.

Let us consider the following are the encrypted secret file bits needs to be hidden in above audio file bits as shown in figure 3.7. The bits in bold are the one’s encountered by moving forward through disposition. The bits in blue are LSB bits of the original audio pattern.

```
"11001010 00101000"
```

Fig 3.6 Sample Audio Bits Showing Disposition Bits

Fig 3.7 Audio Bits After Substituting Secret File Bits
This is the pattern of bits after substituting the LSB of audio by encrypted secret file bits. It can be seen not all LSB are replaced, but only the LSB of those bytes encountered by disposition. In the above image the bold bits are the bits encountered by disposition, and the red colored bits are the secret file bits. Thus the replacement of bits follows a certain pattern depending on the size of audio file size to encrypted secret file size.

*Generate a Stego Output file as a Wave*

The tool first reads the header of the wave file and copy the same header to the stego file it generates. Since the header remains same as that of initial wave, the output file that is generated will be the wave file with same properties as initial wave. The encrypted secret file size is written into LSB of first 32 data bytes of audio. This helps the system to keep track of encrypted secret file size. And thus it can calculate the disposition from audio file size from available wave stego, and encrypted file size available from first 32 bytes of audio data.

Consider that secret file size be 1Kb, which is equal to 10000000000 in binary representation. These bits will be padded by zeros on left to make a 32 bit number as “0000000000000000010000000000”. Consider the following is the first 32 bytes of audio as shown in figure 3.8.
### Fig 3.8 First 32 Data Bytes of Audio File

The following are the first 32 bytes of data of audio after saving the information about size of encrypted secret file as shown in figure 3.9.

| 01100110 | 1011010 | 11010100 | 00101010 |
| 01010100 | 01010100 | 11001011 | 01001010 |
| 11001010 | 01001111 | 00101010 | 01101010 |
| 01101010 | 11101010 | 01101010 | 11100011 |
| 10101001 | 00101110 | 00011101 | 00101011 |
| 11001010 | 01001111 | 00101010 | 01101010 |
| 01100110 | 10110100 | 11010100 | 00101010 |
| 11001010 | 11001111 | 00101010 | 01101010 |
3.5.2 Un-hiding Secret Message

Retrieval of secret message from wave stego proceeds in following steps

1) Select/ retrieve all hidden encrypted secret file bits in audio

2) Using the Key generated during Encryption decrypt the Secret message bits

3) Generate the output as initial secret file

Select/ Retrieve all Hidden Image Bits in Audio

The stego file generated which is a wave file by properties has secret file stored in it. First the tool reads the header of stego file to get size of wave file. Then it reads the first 32 data bytes, which has size of secret file. Now calculate disposition using wave file size and encrypted file size. Since, this number gives the information where the secret
data bits are stored in LSB of wave file, retrieve all these bits. Thus the tool retrieves the encrypted bits of secret file.

**Using the Key Generated During Encryption Decrypt the Secret Message Bits**

Once the system retrieves bits from the audio/wave file are in encrypted form. The decryption technique generates a key from the password user entered during retrieval of secret data. This password entered during retrieval should match the password entered during hiding. TDES follows a reverse process during decryption. The following is the variant of TDES “DES(k3;DES -1 (k2;DES(k1;M)))”. Here k3 = k2 = k1 = k, which is generated based on the password entered by user. Thus the original secret message bits are decrypted to an order.

**Generate the output to original secret file**

The user has to know the extension of the secret file to generate the original message from the decrypted bits. Thus the tool retrieves back the original message hidden in audio.
4. EVALUATION AND RESULTS

The system designed has been implemented and tested in various ways to find accurate results and the testing has been successfully completed. The following are the various tests that have been undergone in the process of testing:

1) White Box Testing
   - Unit Testing
   - Integrated Testing

2) Black Box Testing/ Functional Testing

4.1 White Box Testing

White-box testing, also known as structural or glass-box testing, is performed to reveal the internal code problems. White-box testing uses the procedural design control structure to derive test cases, independently of the program’s requirements. In one standard strategy for white-box testing, the tester creates a set of test cases that exercises each of a program’s points. Thus one can use White-box testing for testing the code structurally, so that internal code problems can be resolved.

The code has been developed for hiding any file into a wave file and retrieving back. The developed code successfully accomplished white box testing and there have been no errors in resulting code.

4.1.1 Unit Testing

A unit test verifies the smallest unit of the software, i.e., software component or module. Thus unit testing can be used to test encryption and decryption methods separately as a unit and then can be combined.
Here the code has been developed as a module basis separately for converting wave and image into bits, developing encrypting algorithm, calculating image and wave file sizes, hiding image bits into wave file during the process of hiding secret data into image. All these modules have been successfully tested similarly.

4.1.2 Integration Testing

The second level in testing, integration testing, assembles unit-tested software into packages and resolves interfacing errors. The individual modules used in hiding process were integrated and tested successfully. This process has been repeated for all modules of retrieval of secret file.

And finally both hiding secret message module and retrieving secret message module are integrated and tested successfully.

4.2 Black-box Testing

Black-box testing also known as functional testing, exercises a program’s requirements. In black-box testing, the tester develops the test cases based on the software’s required functionality. Functional tests typically check that valid inputs produce known or expected outputs; they are designed with no knowledge of the code’s internal logic. Thus the functionalities of the system developed using Black-box testing can be tested.

To test the functionality of this tool, a wave file of 27.5Mb size and a image file of 1.19Mb size have been considered as inputs. After hiding the image into wave file, it has been found that the wave file size (27.5Mb) has remained same. Also once the image has been decrypted, it was noted that all properties of image remained same and there was no distortion in image.
4.3 Validation Testing

Validation testing uses black-box testing techniques. Testing begins with the development of a test plan that outlines the classes of tests to be conducted. A test procedure is designed that specifies test cases to confirm the requirements. According to Pressman, the test plan and procedure “should ensure that all functional requirements are satisfied, all behavioral characteristics are achieved, all performance requirements are attained, documentation is correct, and human engineered and other requirements are met” [Pressman, 2001]. A validation test confirms the functional and performance characteristics to specification. Or a validation test may uncover deviations from specifications and the validation tester creates deficiency list.

Validation testing is depicted below using the figures below, which are screenshots of the testing the tool. Figure 4.1 shows the initial size of the audio song, which is a cover file. Figure 4.2 shows initial size of a secret file, here it is an image file which is depicted in figure 4.3. Once the image is hidden into audio file, a stego file is generated which is shown in figure 4.4, which confirms the size of secret file remains same. Once the image is retrieved back, the figure 4.5 shows the size of secret file still the original, and also figure 4.6 shows the content remained same.

*Initial Size of Wave file/Carrier file*

Fig: 4.1 Initial Size of Wave file/Carrier file
Initial Size of Image before Hidden into Audio/Wave file

Fig: 4.2 Initial Size of Image before Hidden into Audio/Wave file

Image Content before Hiding

Fig: 4.3 Initial Image Content

Wave File Size after Hiding Image

Fig: 4.4 Wave File Size after Hiding Image
Thus the system has passed Validation testing successfully.
4.4 Alpha and Beta Testing

Software is used by a wide variety of customers/users, which makes it nearly impossible for a software developer to foresee how users will use software. The user may misinterpret instructions or use the software to manage untested datasets. These situations may uncover unexpected errors and deficiencies in the product.

This tool has been given for both computer science students and non-computer science students. All the people found the tool user friendly and implemented the wave steganography successfully.
5. CONCLUSIONS AND FUTURE WORK

The “wave stego” tool that has been developed is able to hide any file into audio file and is also able to retrieve back the hidden file successfully without any distortion. The important criterion to observe here is, the simplicity of obtaining the result. Here during hiding, the tool copied the header of initial wave and during un-hiding, it just gets the header of image from first 32 bits of wave data file. This tool has been constrained to a secret message size up to 1/8th of carrier size (because of lsb substitution technique). The encryption algorithm here being used is Triple Data Encryption Standard, which encrypts secret file data before hiding into wave file.

There are many areas in which this tool can be expanded

1) Use of Encryption Algorithms

2) Use of Carrier Files and

3) Size of Secret Files

4) Three keys can be used

5.1 Usage of Encryption Algorithms

Triple Date Encryption Standard algorithm has been used to encrypt the secret message bits to achieve security to data. AES (Advanced Encryption Standard Algorithm) which is more powerful and secure, can be used for encryption in place of TDES.

5.2 Usage of Carrier Files

Here the code developed initially converts carrier file to bits and uses the same header to generate stego file. It will be a relatively easy task to extend the project to make use of a different carrier file such as wma, wmv, mp3 etc.
5.3 Usage of Secret Files

This tool has been constrained to use a secret file up to $\frac{1}{8}$th of carrier file. The hiding pattern can be changed to extend the tool to accompany a larger hidden file to a relatively smaller carrier file than that used in this application.

5.4 Usage of Three Keys

This tool makes use of the password entered and generates one key and encrypts the secret data three times using the same key. The tool can be extended to generate three keys based on the password used and encrypted three times, each time with a separate key.
ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Dr. Dulal Kar, Associate Professor of Computer Science, Texas A&M University – Corpus Christi, for his expertise in area of Network and Information security, for his keen interest in introducing Steganography as a Directed Independent Study, unending support, much needed motivation, and for providing guidance for completion of my project successfully.

My Sincere thanks to Dr. Mario Garcia, Associate Professor of Computer Science, Texas A&M University – Corpus Christi for his unending support, introducing s-tools, and for providing guidance which helped me towards completion of my project.

My sincere thanks to Dr. David Thomas, Associate Professor of Computer Science, Texas A&M University – Corpus Christi, for his keen interest, unending support and warm wishes that provided me the much-needed motivation in completing my project.

And finally heartfelt gratitude to all the faculty, and staff of the Department of Computing Sciences, my colleagues and friends for their moral support.


