Unseen: An Overview of Steganography and Presentation of Associated Java Application C-Hide

Jessica Codr, jmc5@cec.wustl.edu (A project report written under the guidance of Prof. Raj Jain)

Download 

ABSTRACT:

People have desired to keep certain sensitive communications secret for thousands of years. In our new age of digital media and internet communications, this need often seems even more pressing. This paper presents general information about steganography, the art of data hiding. The paper provides an overview of steganography, general forms of steganography, specific steganographic methods, and recent developments in the field. The information presented in this paper is also applied to a program developed by the author, and some sample runs of the program are presented.

KEYWORDS:

steganography, steganalysis, data hiding, data security, data embedding, stego-objects, watermarking, secret communications, secret messages, hidden messages, hidden channel, covert channel, LSB alterations

TABLE OF CONTENTS:

---

1. Introduction and Overview

Have you ever set up code words for talking with your friends so that you could convey something to them without those nearby knowing you were doing so? Perhaps you established a code word or signal to be used at a party to indicate you were bored and ready to go home or, if you are more devious, established a system to cheat at a card game. If you have done anything like this, you have used steganography. Steganography is the art of hiding a message so that only the intended recipient knows it is there. In the most widely cited description of steganography, two prisoners, Alice and Bob, are trying to plan a jail escape while under the watchful eye of Warden Wendy. Wendy will not tolerate suspicious behavior, such as passing notes that are clearly encrypted. So Alice and Bob communicate such that it seems they are talking about something harmless (such as the weather or their families) when they are actually planning an escape (cited in [Bergmair06] from [Simmons84]). From this simple theoretical example, many steganographic techniques and practices have spawned and have helped improve data security in the real world.

1.1. Steganography Versus Cryptology

In the "real world", steganography, like cryptology, is intended to add a layer of security to communications so that pesky eavesdroppers don't know what Alice is saying to Bob. However, unlike cryptology, steganography is not meant to obscure the message, but to obscure the fact that there is a message at all. Attacks against cryptography take what is known to be an encrypted message and attempt to decrypt the message. Attacks against steganography take what seems to be an ordinary image, text, multimedia file, or other document and determine whether or not there is another message hidden within.

Steganography and cryptography are strongest when combined. A message sent in secret (steganography) in an encrypted form (cryptography) is much more secure than a "plain text" message sent by secret means or a clearly sent encrypted message. There are some cases in which steganography can take the place of cryptography; for instance German bans on encrypting radio communications were recently countered by applying steganography to radio communications [Westfeld06]. Generally, however, steganography "is not intended to replace cryptography but supplement it" [Johnson95].

Steganography, like cryptography, also has its own set of terminology. In steganography, cover refers to the media in which a message is hidden. Covertexts and coverimages are texts and images used as covers, respectively. A stego-object is the cover with the secret message embedded in it.

Steganography also has an additional branch known as watermarking, which is a means of hiding data within a cover in order to mark that cover and prevent duplication or unauthorized use. Whereas pure steganography hides data completely, watermarking is meant to be detectable but unalterable. Watermarks can be applied to text documents containing intellectual property, art work, music files, movies, or anything that an author or owner does not want others to use or copy without proper authorization. The watermark verifies a media file

owner's right to use it. If the watermark can be removed, systems that check watermarks to see if the user is authorized to have the media just see an ordinary file with no protection and allow the owner to use it. Thus, watermarks must be "hidden" so as not to damage the media and must be detectable by an outside system, but not removable [Lu05][Katzenbeisser00]. This discussion of the purpose of watermarking leads into a more general discussion of the goals of strong steganography, presented in the next subsection.

1.2. Characteristics of Strong Steganography

Though steganography's most obvious goal is to hide data, there are several other related goals used to judge a method's steganographic strength. These include capacity (how much data can be hidden), invisibility (inability for humans to detect a distortion in the stego-object), undetectability (inability for a computer to use statistics or other computational methods to differentiate between covers and stego-objects), robustness (message's ability to persist despite compression or other common modifications), tamper resistance (message's ability to persist despite active measures to destroy it), and signal to noise ratio (how much data is encoded versus how much unrelated data is encoded). The three main components, which work in opposition to one another, are capacity, undetectability, and robustness. Increasing one of these causes the others to decrease; thus, no steganographic technique can be perfectly undetectable and robust and have maximum capacity [Salomon03]. In most cases, capacity is not as important as the other two, and whereas watermarking favors robustness most strongly, general steganography considers undetectability the most important [Salomon03]. A summary of the properties of good steganography is presented in figure 1 below.

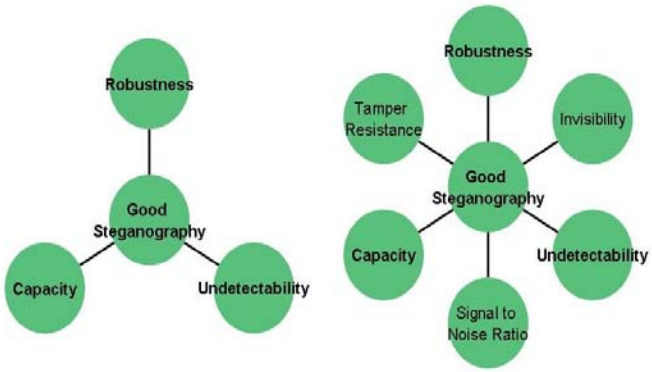

Figure 1 Properties of Good Steganography: the three most simple opposed properties (left) and a display of all six key properties (right)

As some of these properties indicate, steganography seeks to be strong against steganalysis, which is the attempt to uncover the hidden message within a stego-ojbect. Figure 2 summarizes the steganalysis process. Steganalysis can combat steganography in ways other than detecting the message, but determining how to uncover the message is the main problem steganalysis seeks to solve.

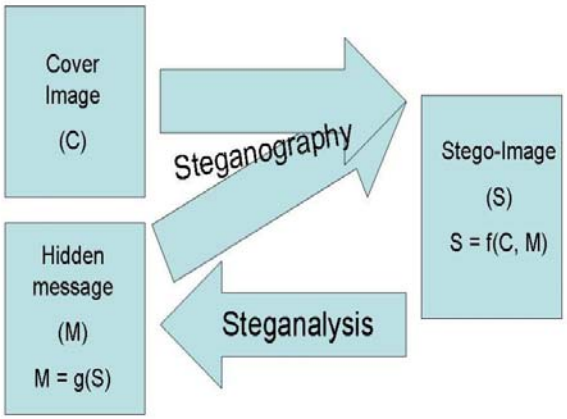

Figure 2 Steganography and Steganalysis

[Kipper04] presents five classes of steganalysis: stego-only, known-cover, known-message, chosen-stego, and chosen-message. In all cases, we assume the attacker has the stego-object. In stego-only, this is all he has, whereas in known-cover and known-message, he also has the original cover without the hidden message and the message itself, respectively. If he already knows the message, the goal is to find how it was hidden. This could, for example, help find future uses of steganography in a stego-only situation. In chosen-stego, the entity (human or computer) trying to find hidden messages knows what algorithm is used to hide messages, and in chosen-message, he is able to use some program or method to embed his own message and use this to find similarities to other possible stego-objects that might indicate the presence of a hidden message. A summary of these five types of steganalysis is given in table 1 below.

Table 1 Steganalysis Categories

No extra information	One extra piece of information	Algorithm testing methods
Stego-only	Known-cover	Chosen-stego
	Known-message	Chosen-message

One weakness of steganography is that its strength often depends on attackers not knowing how data could be hidden [Katzenbeisser00]. Contray to this, strong and successful cryptosystems make their algorithms public knowledge and encrypted messages still can not be broken. Steganography would be strongest if the algorithms used to hide data were known and attackers still could not tell whether or not hidden data was present. Fortunately, steps are being made towards steganography that relies for its security only on the key used to hide the data and not on the method used to do so. This is one way steganography has developed over the years, and we see as we touch on its origins in the next subsection that it has come very far to get to this point.

1.3. Origins of Steganography

Steganography was around long before computers were invented. As long as people have desired to

communicate in secret, steganography has been there, allowing them to at least attempt to do so. The term "steganography" dates back to 440 BC and derives from a Greek word meaning "covered or hidden writing." This word was used to refer to practices of leaders hiding messages sent to other leaders. One early such practice used by the Greeks was to scrape wax off of tablets, write on the wood underneath, and cover the message with the scraped off wax. In another early steganography example, a man's head was shaved, a message tattooed upon it, and the man sent to another leader to deliver the message after his hair had grown back and covered it so that others would not be aware he was carrying a message [Wiki09].

Mention of steganography also appeared in early publications such as "Stegangraphia," written by Johannes Trithemum before 1606 [Salomon03], and steganography tactics were used by historical figures such as Mary Queen of Scots, who sent messages hidden in beer kegs to those hoping to use her to overthrow Queen Elizabeth [Kipper04].

Steganography came to what is now the United States as early as the Revolutionary War, during which it took the form of secret message drops, code words, and invisible inks used for communications between General George Washington and a group of spies [Kipper04]. It continued in use through additional wars, including World War I and II. Following the tragedy of September 11, 2001, investigations revealed that Al'Queda terrorists may have transmitted images containing hidden messages via usenet. Evidence exists primarily in the form of Islamic extremist websites that provide information on how to embed data in images [Kipper04]. Though the use of steganography in planning 9/11 was not confirmed, the possibility of its use sparked new interest in steganography, and led to further research into its use and its prevention.

In the remainder of this paper, I present an overview of cover types, more details about specific methods, recent research advances in steganography, and the java application I wrote to apply this information to real steganography. I begin with a discussion of cover types.

---

2. Cover Media and General Steganography Techniques

As history and technology have evolved, so has the art of hidden communication. Today there are many forms of steganography that use many different mediums to transmit hidden information. Steganography can use essentially any other form of media to transmit a hidden message, though techniques vary from cover type to cover type. Examples of covers that have been used include digital media such as images, video, sound, and executable files; text (via formatting or semantics); and networks and network packets. These general categories will be discussed in overview in the following subsections, and some further details of techniques of interest will appear in section 3.

2.1. Digital Media

The most well-known form of steganography involves hiding messages within pictures. This is often done by altering the low order bits of an image so that the image looks unchanged, but a diff of the altered image with the original reveals a pattern corresponding to a hidden message. This form of steganography has most commonly employed bitmaps due to their simplicity of data representation (when simply flipping the low order bit, the image itself remains completely unaltered visually), but more recent research and work has been done involving the alteration of JPEG images using software such as JSteg to hide messages within seemingly ordinary photographs. JPEGs are trickier to alter due to their "layers" and embedding of data that makes them more complicated than the raw data format of bitmaps [Guillermito05]. More discussion of image based steganography, and specifically the use of JPEGs, will come in later subsections (specifically 3.1, 3.2, 3.3 and 4.2).

One specific application of steganography in images is the watermarking of medical records and images with

embedded patient data so that such data is readily accessible but still secure. The detection of this secret data could also verify that medical files and images are authentic and unaltered [Kipper04].

In addition to images, other media may hide messages. Videos may use individual frames that flash by too quickly to be noticed, but contain secret information when viewed as still images. Video images could also be modified in essentially the same ways as still images, and since video files are larger than single image files, more data could be hidden. Audio recordings may contain background "noise" that actually contains a message, or could be altered in a way that, like the alteration of images discussed above, leaves the sound unchanged, but reveals a message when compared against the original audio file.

Even executable files or file systems could hide data. Unused portions of either could hide significant data. The way files are organized could represent a message, or extra disk space could be used to store what looks like an executable or system file but is actually a secret message.

Normally, especially with steganography that embeds data within images, the cover used is irrelevant: any file of the proper type will do. However, there is also something to be said forselecting a cover most fitting for the intended hidden message. In 2002, for instance, Xu and Feng presented the idea of changing how data is embedded in an audio file based on the actual audio content [Lu05]. Using text to embed data is another example of how the actual cover media can sometimes be relevant to the hidden message, as we see in the next subsection.

2.2. Text

Text's history extends further back than digital media, and thus its use in steganography may be more prominent in older times than that of images. Invisible inks during the U.S. Revolutionary War, Turning Grilles during World War I, and microdots and code books during World War II were all key ways that text has been used to convey secret messages. The Turning Grille was a piece of paper with holes punched into it such that when the device (grille) was placed over a newspaper, the letters visible through the holes revealed a message. A grille could be constructed and sent to the recipient with the knowledge that it must be used on a newspaper to reveal a message [Kipper04]. The microdot was microscopic text embedded within another text, such as within a period at the end of a sentence, too small to be seen by the naked human eye. The discovery of German microdot usage prompted J. Edgar Hoover of the FBI to call microdots, "the enemy's masterpiece of espionage" [Johnson95]. As for the use of codewords during World War II, even such seemingly harmless devices as crossword puzzles were suspected of use in transmitting sensitive military information.

In addition to these historical examples, other patterns of letters within words can reveal secret messages via what is known as a null cipher [Kipper04]. For example, a seemingly harmless message could be written such that by taking the ith letter of each word in the message, another message is revealed. For instance: "Around July, anyone may encounter odd white lights adorning sea-skies" embeds the secret message "Run and hide." To see how, and to find out what message is hidden in the following paragraph, refer to Appendix B.

Hidden messages could also appear in the form of miniscule typeface, size, or spacing differences. Extra spaces before certain words could indicate that those words or the first letters of those words should be taken apart from the entire message to reveal a secret embedded utterance. This is especially handy in html files since extra spaces show up only in the source file and not on the webpage display. Letters that are slightly larger might similarly be taken to reveal a hidden message. It could even be that, through use of invisible ink between lines of text or tiny print within underlining or punctuation, the true message is not visible at all. Some of these methods may be easier to detect than others, but they have had their own practical uses in history, as we will saw in the previous section. Can you find the message hidden in this paragraph?

The idea, previously presented, of making the cover an integral part of the message hiding process is even

more applicable to the use of text as a cover. Beyond simply changing methods based on the cover, a covertext can be tailor-made with the hidden message in mind. For example, there exists a program called NICETEXT that generates seemingly "normal" messages based around some desired hidden message [Salomon03]. NICETEXT can generate email messages that appear to be SPAM mail, but also bear the secret message [Kipper04]. The email can then be sent to one or several people, including the other communicating party who knows how to look for the hidden message. This form of secret communication relies on network communication, and as we shall see in the following subsection, there are additional forms of steganogaphy that rely on networks even more directly.

2.3. Network Communications

The combination of text with network communications is an excellent way to communicate secretly in modern society. Not only does email enable sending hidden messages, but the net as a whole can be used for secret communication. Posts on forums could be engineered to communicate secretly between different groups. The phrasing of a post might vary based on the message one wishes to send, or the message board that is posted on could be meaningful. What seems like a typical forum response could contain secret data.

Apart from words and media, the manner in which data is transmitted could convey a hidden message. By arranging to deliberately drop certain packets or change the bandwidth of a channel at certain times, one can send a message to another party undetected by those who don't know to look for it. To an outside observer, dropped packets simply look like packets dropped due to unreliable data channels. But to one who knows what to look for, a pattern formed by those dropped packets, such as from how often they are dropped, which ones are dropped (see figure 3 below), or their retransmission times, could actually be an encoding of a secret message that no one else even knows to look for.

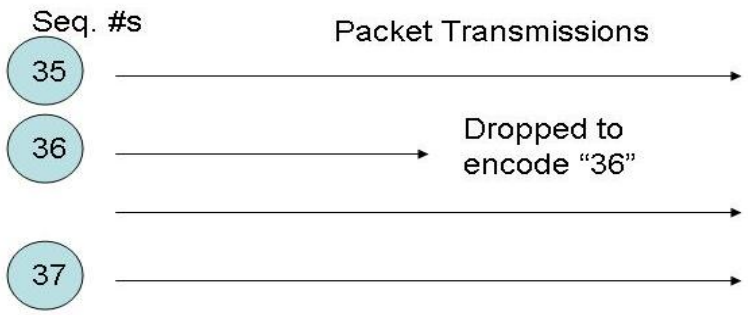

Figure 3 Sample Use of Network Steganography

Unused headers in IP packets can also hide data. In IPv6 in particular, there are at least 22 ways packets can be used to send secret data [Lewandowski06]. More discussion of the use of IPv6 headers in steganography and the countermeasures developed against this are given in the Cutting-Edge Developments section (section 4), which follows the next section on specific steganographic tactics. This following section will delve deeper into specific techniques used to hide data within other data.

---

3. Specific Steganographic Tactics

Extending from the general tactics described above are many specific techniques used to hide data within a cover. The specific tactics can vary based on the type of cover and which of the steganographic characteristics (from section 1.2) the technique seeks to emphasize. Each technique has its own strengths and weaknesses, though some techniques are widely considered better than others. This discussion begins with

one of the simplest, and weakest, techniques available.

3.1. Least Significant Bit Alterations

The most basic way to hide data within an image file, as already discussed, is in altering low order bits, which modifies color data slightly, but not in a way visible to the human eye (see section 5.2 for some visual examples). This is called a least significant bit (LSB) alteration or modification. The equivalent when using audio files is called amplitude modification [Lu05]. Computers can subtract the original image from the altered image to reveal a message or even another picture. There are many different ways to use these alterations to hide a message. Perhaps writing out the LSBs in order reveals the message or every alteration of an LSB indicates a "1" and whereas each time the LSB is unchanged a "0" is indicated. These kinds of modifications to individual pixels in an image or to data units in some other cover are called spatial domain modifications [Lu05].

Now it is easy to add a hidden message undetectable to human, but trick is making it undetectable by a computer performing steganalysis. By altering LSBs, we create a pattern a computer can see even though a human can't. A computer can easily detect differences between the original and altered image if it has access to both as this is the way the message must be revealed to those who know to look for it. This makes steganography highly susceptible to detection by someone who has access to original images.

Even without the original images, bit patterns can possess unexpected irregularities that reveal the presence of a message. Normally, low order bits of an image may have random alterations in them or may vary based on the image itself. Solid blocks of the exact same color are often the same even in their low order bits. By using steganography to alter these bits in a systematic manner, we create an abnormal pattern that a computer could pick up on even though a human eye could not.

If you detect irregularities with a computer, you can enhance the LSBs to see clear abnormalities when text is encoded into the image, even without comparing to the original image [Guillermito05]. This is especially easy to detect if the values of the bits directly translate to the hidden message. Statistical analysis of the bit patterns in the image or other media compared to the expected types of patterns is especially helpful in discovering the existence of hidden data [Katzenbeisser00].

One counter to this statistical analysis is to embed data using Gray Codes instead of direct binary. With Gray Codes, numbers are still encoded with 1's and 0's, but each adjacent base 10 digit differs from its neighbors by only one binary digit. This leads to more uniformity in pixel alterations and causes the LSB values to seem more normal because they are more similar and follow more of an expected pattern [Salomon03]. Another option to try to improve security is to encode the data somewhere where it is less expected, such as in a color lookup table instead of directly in the image [Salomon03]. Even with tactics like this, however, LSB alteration is considered the weakest form of steganography, and much research has gone into alternate approaches, such as those discussed in the following subsections.

3.2. Transform Domain Techniques

There are alternatives to the relatively na&#65533;ve and detectable LSB approach. Specifically, instead of hiding data directly in the least significant bits of the final image, we hide it within some sort of transform of the file, for example one that might be used in compression of the file (for example, in JPEGs). Such alterations are said to be part of the transform domain [Lu05].

These transforms create a hidden message that could be harder to detect, and one that could be more robust against further transforms and modifications of the file because the message is built in as an integral part of such transformations. However, these methods also have the potential to be less robust against alternate types

of transformations because the data embedding is such an integral part of the transforms of the cover that altering these transforms could lead to the lose of more than a few bits of the hidden message [Lu05].

The three main types of transforms used for steganography are the discrete cosine transform (DCT), the discrete Fourier transform, and the wavelet transform. These methods all have coefficients associated with them that define how the image or file should be transformed, and it is within these coefficients that the secret data is hidden. The DCT method involves supplying the image or file to a function (the discrete cosine transform) that outputs coefficients which are then modified to embed data. The variations of this technique that are most wide-spread were first introduced in 1995 by Cox et al and by Koch and Zhao [Lu05]. As of the writing of Lu's book in 2005, the wavelet transform was the most efficient method [Lu05].

Other ways of embedding data, and specifically of spreading it more uniformly throughout the cover, have also been presented in more recent years, as we see in the next subsection.

3.3. Data Dispersal and Feature Modification Techniques

It was often also thought to be more undetectable to spread the hidden message over several covers, though recent research indicates this may not be true (see subsection 4.4). Nonetheless, methods such as Patchwork, developed by Bender in 1996 and described in [Salomon03], seek to embed just one bit of data per image and spread the entire message over many images. Patchwork changes the luminance of two pixels in opposite directions slightly enough that the change is not detectable, but still enough that the difference in luminance between the two pixels is significant and indicates an encoded bit of data.

Rather than going to the extreme of spreading data over entirely separate covers, one can also spread data across pieces of the same cover. Instead of hiding data in individual bits, one can hide it in blocks of pixels. For example, a purely black and white image could be divided into 8x8 blocks of pixels and the number of white pixels compared to the number of black pixels in each block could represent secret data (eg more black = 1, more white = 0) [Salomon03][Lu05]. One weakness is that it is easy to "crack" this encoding if the method is known, or to alter the embedded message (meaning the method lacks robustness). This approach is an example of the general class of feature domain modifications and embeddings. Working with the feature domain means embedding data in regions, boundaries between regions, or within the objects appearing in a cover [Lu05].

One strength of feature domain embeddings is that alterations to the overall geometry of a cover may have less of an effect on such methods [Lu05]. Another feature domain modification technique known as "geometric warping" involves intentionally warping or shifting some features in video images in a way that embeds information [Profrock06]. Though the images are actually changed (eg shifting a house over), humans don't notice because the image still looks okay. Additionally, the changes are computationally significant and thus will not be compressed out by standard compression methods or other image modifications.

Steganography that hides data in unimportant parts of a file, such as LSB methods, can be erased by compression methods that get rid of those unimportant parts [Katzenbeisser00]. Hiding the data in significant parts of the image (discussed in this section), or within the compression algorithm/method itself as is done with transform encoding (section 3.2), counter this.

This wraps up most of the techniques applicable to images, but in this final subsection on specific steganography algorithms, we look at techniques applicable specifically to non-image data.

3.4. Non-Image Techniques

Although the same basic steganographic ideas can be applied to many types of files, the specific approaches

must change based on file type. For instance, audio and image files are treated very differently and specific techniques such as adding echos to an audio file to hide data do not apply to image files [Lu05]. Another technique that applies to audio files specifically is phase coding, which hides data within the phase differences between separate chunks of the sound file [Lu05].

Other methods that apply more to other types of communications than to image files include spread spectrum and masking. With spread spectrum, a narrow band of communication hides within a larger one. This applies especially to radio communications and can be used to counter jamming of signals. The idea is that the signal wave is widened and noise that seems random is added, but each section of noise actually has a part of the secret message embedded within it. The two general types of spread spectrum are known as direct sequence (DS) and frequency hopping (FH) modulation [Furht05].

Masking, on the other hand, seeks to cover up the secret data completely with other data or noise rather than just embedding bits and pieces of it within noise. A very direct, literal example is using loud sounds (such as music or running water) to veil a whispered conversation from an eavesdropper [Kipper04]. This type of tactic seems pretty basic and well understood, as may some of the other techniques presented in this overall section. In the next section, I present cutting-edge developments in steganography tactics, including entirely novel ideas about how to use steganography to ensure data security and how to counter steganography when it is misused.

---

4. Cutting-Edge Developments

Steganography is a continually advancing field, and researchers are always coming up with ways to improve steganography, steganalysis, and watermarking. The main event for discussion of steganography and watermarking techniques is the Information Hiding (IH) conference. Most of the developments discussed in this section draw from papers presented in the somewhat recent IH 2006, the 8th International Information Hiding Workshop.

4.1. Novel View of Steganography

Perhaps the most intriguing paper of the 2006 IH workshop presents an entirely novel approach to how researchers should view steganography [Bergmair06]. In the past, steganography has hidden data in unimportant parts of files and altered the "syntax" of covers without being concerned with "semantics." The goal was to hide messages from humans who did not know to look for them, and this goal was achieved quite well. However, computers were very good at detecting the secret communications.

The revelation of this paper is that most steganalysis attacks today come from computers, so it makes more sense to focus on fooling machines than to focus on fooling men. Though this might not always be desired, it allows for communications to more often go immediately unnoticed since computers (not humans) are usually used to immediately detect secret communications. The paper proposes taking advantage of the computer's inability to solve very simple "puzzles" that are obvious to humans but not to current artificial intelligence (AI) tactics. The two examples given are image labeling and grammars involving sets of synonyms.

In the first case (illustrated in figure 4 below), the human recipient detects which of a series of textual labels for different images is actually incorrect and the position of this image in the list of images embeds part of a message. This method can be repeated multiple times to reveal a full message.

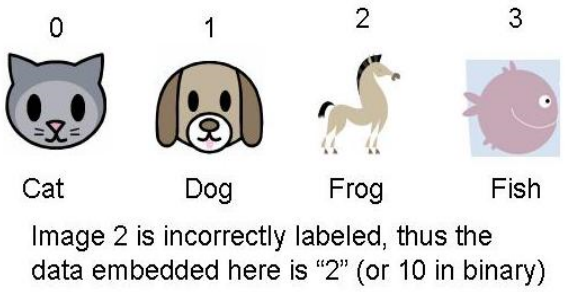

Figure 4 Steganography via Image Labeling

In the second case, a word in a sentence is substituted with a synonym that seems syntactically and possibly even semantically correct to a machine, but which a human can tell is being used in the wrong context. The index of the "incorrect" word within a predetermined set of possible synonyms indicates part of a hidden message. In both cases, a computer cannot do what a human can do to detect the "problem" that indicates the presence of a hidden message and thus remains blissfully unaware of any cause for concern [Bergmair06]. The novelty of this idea might be a bit too extreme for some steganographers, and so there are many other new advances that remain within the realm of the old views of steganography. These are presented in the following subsections.

4.2. Advances with JPEGs

Using images to embed data is a well-established steganography practice, but it was long believed that embedding data in JPEGs was extremely difficult. As previously mentioned, JPEGs are harder to use in steganography than bitmaps because JPEG image data itself is encoded instead of being presented as raw red-green-blue (RGB) values, so a steganographer must be careful not to alter the data in such a way that the image appears altered and thus gives away the presence of a message.

Even when the image itself is not visually altered, though, it can still be hard to hide a message such that other computers are not able to detect its presence. To fight this, the Modified Matrix Encoding Technique adjusts "errors" in JPEGs to encode messages while maintaining the expected distributions of bit values [Kim06]. Basically, some degree of alterations or "errors" in low order bits are expected in digital media, and by spreading out the alterations introduced by steganography, one can keep a computer from determining that there is a hidden message unless it knows exactly how to look for said message.

Modified Matrix Encoding is believed to be unbreakable by current steganalysis techniques, unlike the previously strongest JPEG steganography techniques of OutGuess and F5. Modified Matrix gains some of this strength by "reducing [the] capacity or payload and spreading the message across the whole carrier" [Kim06]. This technique sheds new light on using JPEGs in steganography, but JPEGs are not the only cover media getting more attention in recent years.

4.3. Advances with Networks

As the internet continues to grow in use, steganographers turn even more attention to how communications through this medium can be used to transmit hidden data. In a method that requires use of a network, researchers present the idea that the reordering of packets can be used to convey secret information. This method relies on using a packet transmission protocol that uses sequence numbers and orders packets, such as TCP [Chakinala06]. A separate paper also presents new techniques involving the timing of packet transmissions and how this can be used to hide data [Martin06].

These are two of several methods that can be used to transport hidden data via network communications.

Another paper from IH 2006 discusses ways to counteract such methods used in IPv6 [Lewandowski06]. There are no fewer than 22 different ways that IPv6 can transmit hidden data, including hiding data in listed IP ranges, routing headers, hop limits, or TTL fields. Lewandowski's paper presents the idea of an "active warden" that checks the fields in the packet headers and normalizes traffic when things seem out of place. The goal is to eliminate hidden data without damaging non-hidden data. The methods presented are able to counteract only 2 of the 22 data transmission methods examined, however, so much work remains to be done [Lewandowski06]. Other methods of defeating other forms of stegonagraphy, discussed in the next subsection, have proven more productive.

4.4. Steganalysis and Artificial Intelligence

Steganalysis is fundamentally a problem of classifying samples as either simple covers or stego-objects. This binary classification lends itself well to artificial intelligence methods such as support vector machine techniques, as presented in another paper from the 8th International Workshop on Information Hiding. Results from using support vector machines to detect steganography in JPEGs show that this method outperforms steganalysis techniques developed by other researchers to combat some of the most advanced steganography techniques used on JPEGs: OutGuess, F5, and model-based stego MB1 [Shi06]. The fact that other researchers such as Jessica Fridrich were already able to break these techniques, however, may indicate they were not especially strong to begin with [Fridrich06].

Similar to other AI-like problems, steganalysis benefits from having many samples to examine. The more files we have to look at that might or might not contain hidden messages, the more likely we are to find how data is hidden or whether it is hidden at all. This process of looking at many different files for hidden data is known as pooled steganalysis [Ker06]. With pooled steganalysis, the more files that actually have data hidden in them, the better the steganalysis is at detecting the data. Thus, contrary to what was previously thought by steganographers, batch stegangography, that is, spreading hidden data over many different covers, makes the steganography more vulnerable to attacks [Ker06]. As we see in the next subsection, it is not even necessary to be able to detect hidden messages, simply destroying the hidden messages may be a sufficient attack against steganography.

4.5. Other Recent Developments

One important modern realization is that steganography can be very dangerous, but is not nearly as dangerous if the hidden message is obscured so that it can no longer be read by those to whom it is sent. The practice of "steganography-jamming" is meant not to detect and uncover messages, but to ensure that if a hidden message is present, it is destroyed. In the technique known as, "double-stegging" one applies steganography to an image with hidden text in it already to obscure the message. This counter measure was developed by Bertolino, a recent college grad [Adee08].

Though this specific approach is new, the idea of countering steganography by destroying the hidden data is not. Common ways to attempt to do this include blurring, noise addition, noise reduction, sharpening, rotating, resampling, and softening [Kipper04]. As of five years ago, steganography could not maintain data robustness against these attacks, and though it is still difficult to maintain robustness, advances in other steganography characteristics, such as capacity, have developed in recent years.

Data capacity, that is, how much data can be hidden in a cover, is one of the evaluation factors for steganography (see subsection 1.2). A recent advance in this vein is the use of 3D meshes as covers such that the distances between vertices in the mesh embed data [Wu06]. This method is also robust against changes to the file that don't actually change its content since the data is embedded directly within this content.

In general, there is a strong desire in modern steganography to make LSB manipulations less noticeable and

cause less distortion in data. One thought is that the best way to do this is to minimize the average "distance to code," that is, the number of changes to the cover [Fridrich06]. The paper on this topic states, "Matrix embedding using linear codes (syndrome coding) is a general approach to improving embedding efficiency of steganographic schemes" [Fridrich06]. Such techniques employed include binary +/- 1 embedding aka LSB matching and ternary +/- 1 embedding. More details of this all the methods discussed in this section can be found in papers from the 8th international IH Workshop.

This paper now presents a sample application of steganography to image files, based on some of the techniques and ideas presented, and concludes with a summary of all the techniques and information presented in the paper.

---

5. Author's Application and Conclusions

Taking all of this information into consideration, I created a simple but somewhat novel steganography tool for embedding data in image files. Here I discuss the state of the program and, briefly, potential plans for future work.

5.1. My Application: Description

My steganography program, which I call C-Hide after the initial of my last name, uses basic LSB altering (discussed in section 3.1) with a relatively low data capacity. Specifically, the capacity is at most the number of rows in the image as each character of the plaintext is embedded in a separate row. The method works by dividing each row into chunks of columns with each chunk corresponding to a different letter (punctuation and spacing is ignored). To encode the letter, a random pixel within the letter's chunk of columns is chosen and its LSB flipped (eg from 0 to 1 or from 1 to 0). The result is a visually unmodified image as well as a file that has roughly the same number of LSB 1s and 0s as it did before the modification. The program also provides the option of adding noise to an image before embedding hidden data in an effort to make it more difficult to separate covers from stego-objects.

Additionally, the program allows the generation of purely "random" images in which to hide data. If the image itself is entirely random, it is more difficult to detect alterations to the image. However, a random image may arouse suspicion and so the option also exists to make the images generated slightly less random.

These "quasi-random" images are generated such that pixels next to each other in the image are similar in color. The result is something that could, for example, pass as a form of modern art, which would be slightly less suspicious to humans observing them from the outside. The random image generator itself can also be passed off as an experiment in computer-generated art, which is in and of itself quite interesting, thus hiding the true intention of its use in steganography.

For more on the design decisions for this application and on how to use the application, please see the associated documents [Codr09]. The design decisions document is also reproduced below in appendix C. For an evaluation of some results from running the program, proceed to the next subsection.

5.2. My Application: Results and Evaluation

Especially after reading Guillermito's discussions of weaknesses of older steganography tools, I am keenly aware of how difficult it can be to make a simple but difficult to detect embedding of messages within images, particularly when such encodings only involve changing of LSBs. Guillermito ranked most of the steganography software he examined as relatively poor, and I tried to get around some of the faults he found

by spreading out the data and encoding very little of it [Guillermito05]. To counter statistical counter measures, I specifically tried to make the counts of 1's and 0's appearing as LSBs and how often these values changed within the image to be nearly the same both before and after the image was altered.

The following images (figures 5-8) show some results from running my program on a photo of a cat (courtesy of M. Bensley) and on my randomly generated images. The message encoded in each case was: "We will attack at dawn. Pass this message on to your generals and to no one else. I add even more to confuse the matter further."

The statistics given for each pair of images show the number of even and odd pixels before and after alterations and also the number of times pixels change from being even to odd or visa versa, which I denote as "switching pixels." You can see that the total counts are very similar both before and after. Note that the sum of the differences is less than the total number (100) of letters in the hidden message.

Figure 5 Photo Image Test

Even pixels: 136827 original and 136841 encoded

Odd pixels: 136785 original and 136771 encoded

Switching pixels: 135314 original and 135326 encoded

Figure 6 Photo Image With Added Noise Test

Even pixels: 136193 original and 136185 encoded

Odd pixels: 137419 original and 137427 encoded

Switching pixels: 137050 original and 137048 encoded

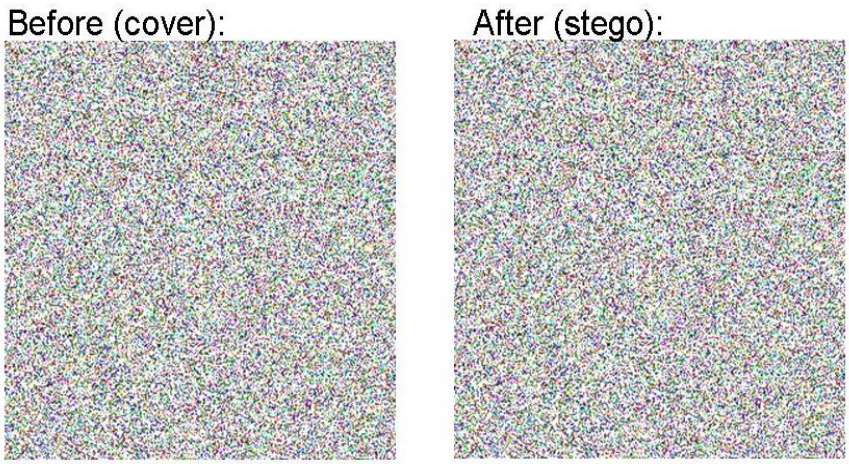

Figure 7 Fully Random Image Test

Even pixels: 22579 original and 22623 encoded

Odd pixels: 67421 original and 67377 encoded

Switching pixels: 33883 original and 33951 encoded

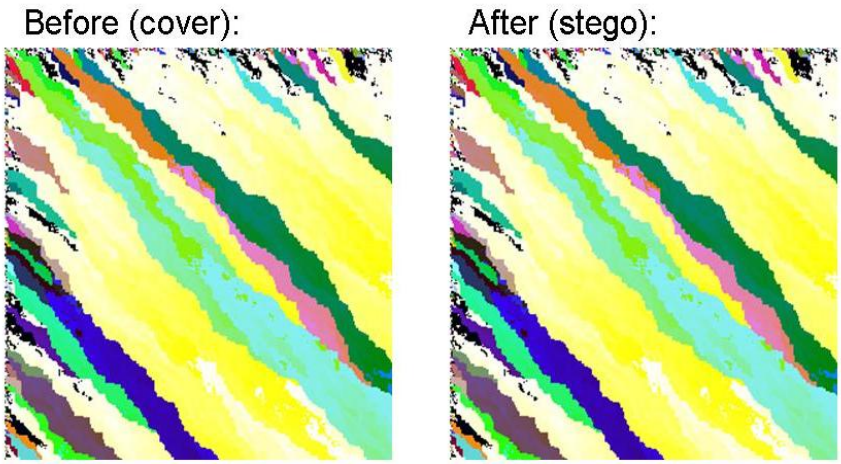

Figure 8 Quasi-Random Image Test

Even pixels: 45092 original and 45092 encoded

Odd pixels: 44908 original and 44908 encoded

Switching pixels: 44875 original and 44871 encoded

From these results, we see that the images appear completely unaltered to the naked eye, and are statistically quite similar as well, making it very difficult for either a human with both images or a computer with only one image to detect something amiss. One observation is that adding noise can improve statistical similarity as the

images in figure 6 are more similar than those in figure 5.

Note also that for the sample pseudo-random image, the number of even and odd pixels is exactly the same before and after alterations. The beauty of randomly generated images is that a user can continue to generate images until one is found with satisfactory statistics or looks, and this often takes only a few iteration to accomplish. This application presents a real and successful application of the ideas presented in the rest of this paper, ideas which are summarized in the following final subsection.

5.3. Final Conclusions

In this paper, I presented an overview of steganography starting with definitions and basic principals and proceeding through cover media types, specific techniques, cutting edge developments, and finally my own application of steganography in a Java program that embeds data in image files. Steganography was described as the art of communicating secretly, which, for example, prisoners Alice and Bob might use to keep their escape plans secret from Warden Wendy. Readers saw that steganography had its origins in 440 BC and has been used in various historical settings ever since. As the years have passed, steganography has developed as the technology available to implement it has grown.

Many forms of covers exist for hiding messages, from image, video, and sound to text to IP packets. There are many specific techniques for embedding data within these various mediums, and each has its own strengths and weaknesses. Some, such as LSB encoding, are considered especially weak, whereas transform domain and feature modification techniques may be slightly stronger. Other cutting edge developments are emerging to create new methods to both hide and uncover data, and even to completely rethink the way steganography is used. Researchers have contemplated developing steganography that hides messages from computers rather than hiding them from humans, and have developed more advanced techniques for hiding data within JPEGs and within Internet traffic. New research has also explored the role of artificial intelligence in steganography and steganalysis.

Taking all of this data and rich history into consideration, I have developed my own program for embedding data into images. Though it uses a relatively simple and weak form of steganography, I consider it to be interesting and less detectable that other primitive uses of LSB alterations due to its significant addition of randomness into the data hiding process. Future work could involve continuing to develop the randomness of this software and enhancing it with more interesting ways to hide data within images without Warden Wendy being able to detect that the data is there. For more on this specific application, please see the associated document "A Java Application for Data Hiding in Image Files." [Codr09] available in part in appendix C.

References

1. [Codr09] Codr, Jessica. "A Java Application for Data Hiding in Image Files" Associated paper, Author's personal files and Appendix C.
2. [Wiki09] "Steganograpy", http://en.wikipedia.org/wiki/Steganography, wikipedia page on steganography, includes links to many other sources.
3. [Kipper04] Kipper, Gregory. Investigator's Guide to Steganography. Auerbach Publications: Boca Raton, 2005.
4. [Lu05] Lu, Chun-Shien. Multimedia Security: Steganography and Digital Watermarking Techniques for Protection of Intellectual Property. Idea Group Publishing, Hershey, 2005.
5. [Bergmair06] Bergmair, Richard and Stefan Katzenbeisser. Content-Aware Steganography: About Lazy Prisoners and Narrow-Minded Wardens. 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers, Springer, 2006, p. 107-123.
6. [Adee08] Adee, Sally. Spy vs. Spy, August 2008, http://www.spectrum.ieee.org/aug08/659 3, online

article about steganography's uses for conveying information
7. [Salomon03] Salomon, David. Data Privacy and Security. Springer, NewYork, 2003.
8. [Chakinala06] Chakinala, R.C. et al. Steganographic Communications in Ordered Channels. 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers, Springer, 2006, p. 42-57.
9. [Martin06] Martin, Keye and Ira S. Moskowitz. Noisy Timing Channels with Binary Inputs and Outputs. 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers, Springer, 2006, p. 124-144.
10. [Lewandowski06] Lewandowski, Grzegorz, Norka B. Lucena, and Steve J.Chapin. Analyzing Network-Aware Active Wardens in Ipv6. 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers, Springer, 2006, p. 58-77.
11. [Wu06] Wu, Hao-tian and Yiu-ming Cheung. A High-Capacity Data Hiding Method for Polygonal Meshes. 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers, Springer, 2006, p. 188-200.
12. [Ker06] Ker, Andrew D. Batch Steganography and Pooled Steganalysis. 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers, Springer, 2006, p. 265-281.
13. [Westfeld06] Westfeld, Andreas. Steganography for Radio Amateurs- A DSSS Based Approach for Slow Scan Television. 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers, Springer, 2006, p. 201-215.
14. [Fridrich06] Fridrich, Jessica, Petr Lisonek and David Soukal. On Steganographic Embedding Efficiency. 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers, Springer, 2006, p. 282-296.
15. [Kim06] Modified Matrix Encoding Technique for Minimal Distortion Steganography. 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers, Springer, 2006, p. 314-327.
16. [Shi06] Shi, Yun Q., Chunhua Chen, and Wen Chen. A Markov Process Based Approach to Effective Attacking JPEG Steganography. 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers, Springer, 2006, p. 249-264.
17. [Profrock06] Profrock, Dima, Mathias Shlauweg, and Erika Muller. Video Watermarking by Using Geometric Warping Without Visible Artifacts. 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers, Springer, 2006, p. 78-92.
18. [Guillermito05] Guillermito "Analyzing Steganography Software (for the fun of learning about it)" http://www.guillermito2.net/stegano/ A webpage discussing many different steganography programs from 2002-2004 and their various weaknesses.
19. [Furht05] Furht, Borko and Darko Kirovski. Multimedia Security Handbook. CRC Press: Boca Raton, 2005.
20. [Katzenbeisser00] Katzenbeisser, Stefan and Fabian A. P. Petitcolas (eds). Information Hiding: Techniques for Steganography and Digital Watermarking. Artech House, Boston, 2000.
21. [Johnson95] Neil F. Johnson. Steganography. Technical Report. November 1995 http://www.jjtc.com /pub/tr_95 _11_nfj/sec401.html.
22. [Simmons84] Simmons, G.J. The prisoners' problem and the subliminal channel. Advances in Cryptoloy, Proceedings of CRYPTO 83, 1984, pp 51-67.

Figure 4 clip art courtesy of Microsoft clip art.

For additional reading on new technologies not directly addressed here, see the remaining papers from: Camenisch, Jan L., Christian S. Collberg, Neil F. Johnson, and Phil Sallee (Eds) 8th International Workshop, IH 2006, Alexandria, VA, USA, July 2006, Revised Selected Papers., Springer 2006.

Acronyms

- AI: artificial intelligence
- DS: direct sequence
- DCT: discrete cosine transform
- FH: frequency hopping
- LSB: least significant bit
- JPEG: Joint Photography Experts Group (the image encoding standard created by this group)
- IH: Information hiding
- RGB: red-green-blue
- TTL: time to live

Appendix A: Additional Terms

- carrier: The file or signal use to hide and transmit the payload. (see cover)
- channel: The input into the steganography system, such as an image.
- cover: The media object in which the hidden message (payload) will be transmitted. "Cover" refers to this piece of media before it is embedded with the secret data. Such images are specifically called "coverimages" and texts are specifically called "covertexts"
- double-stegging: A specific stegnography-jamming technique that involves applying steganography to a media object thought to be a stego-object with the intention of destroying the originally embedded message.
- plaintext: An unembedded, unencoded message.
- payload: The message to be transmitted.
- stego-object: The media object after the hidden message has been embedded within it. Called "stegotext" when embedded in text and "stegoimage" when embedded in an image.
- steganography-jamming: A general steganalysis tactic whereby the steganalysist seeks to destroy the hidden message without actually uncovering it.

Appendix B: Puzzle Solutions

In the first puzzle cited in 2.2, "Run and hide" is embedded by taking the second letter of each word in the sentence, as seen in figure 9

Figure 9 Second Letter Stego

"Made it out. Send money" is embedded by taking the first letter of every word that follows an extra space. If viewing this paper as html, the source must be examined to find these extra spaces. Figure 11 reveals the message by underlining the extra spaces and bolding the letters following the.

Hidden _messages could also _appear in the form of miniscule typeface, size, or spacing _differences. _Extra spaces before certain words could _indicate that _those words _or the first letters of those words should be taken apart from the entire message to reveal a secret embedded _utterance. _This is especially handy in html files _since _extra spaces show up only in the source file and _not on the webpage _display. Letters that are slightly larger _might similarly be taken to reveal a hidden message. It could even be that, through use _of invisible ink between lines of text or tiny print within underlining or punctuation, the true message is _not visible at all. Some of these methods may be _easier to detect than others, but they have had their own practical uses in history, as we will saw in the previous section. Can _you find the message hidden in this paragraph?

answer: **MADE IT OUT SEND MONEY**

Figure 10 Extra Spaces Steganography

Appendix C: Java Application Design Choices

The following is a reproduction of the full associated paper: "A Java Application for Data Hiding in Image Files: Design Choices" by Jessica Codr, the author of this paper.

In the associated paper "Unseen: An Overview of Steganography and Presentation of an Associated Java Application," I give a brief overview of a Java application I am developing for embedding secret messages within image files. Here, I offer a bit more in-depth discussion of the reasons for my technique choices and the current capabilities of this application.

First of all, my steganography program uses a form of basic LSB altering. This form of steganography is believed to be weak and easily breakable, but I chose to use it for two main reasons:

1. It is simple so I can easily do numerous experiments related to it.
2. I wanted to test whether I could strengthen this weak technique by adding more randomness.

The method itself does still encode data by flipping LSBs, but I hope that by adding randomness to images and to how data is encoded, I can make the cover images and stego-images seem more similar.

There are several types of randomness that I employ in my application. First of all, the exact pixels altered to encode each letter are semi-random. When embedding a message within an image, the first step is to count the number of letters in the message and divide the total number of rows of pixels in the image by that number. For example, if the image is 1000 pixels tall and I want to embed a message 100 letters in length, I divide 1000 by 100 to get 10 rows per letter. This is the number of rows of pixels that could be associated with each letter. Next, I take the number of pixels the image is wide and divide by the number of letters in the alphabet (26). So if the image is 780 pixels wide, I divide by 26 to get 30 columns per letter of the alphabet.

By grouping these sets of (10 in the example) rows and (30 in the example) columns together, I get a grid of sets of pixels within the image. Each of these subgrids corresponds to an encoding of a different letter of the message. The groups of rows correspond to the index of the letter in the message (eg, first letter, second letter, etc) and the groups of columns correspond to the actual letter encoded. So, for example, to encode the third letter as the letter "b" I would look at the third set of rows (rows 20-29 in our example) and the second

set of columns (columns 26-51 in our example). Now to actually encode that the third letter in the message is "b", I take a random bit from this subgrid (from rows 20-20 and columns 26-51 in our example) and flip it. See table 1 below for a visual representation of how subgrids map to different letters in the message for part of the image.

Table 2 Encoding Locations for Parts of an Image File

IMAGE GRID	Cols 0-25	Cols 26-51	Cols 52-77	Cols 78-103	Cols 104-129
Row 0-9	Letter 1 = a	Letter 1 = b	Letter 1 = c	Letter 1 = d	Letter 1 = e
Row 10-19	Letter 2 = a	Letter 2 = b	Letter 2 = c	Letter 2 = d	Letter 2 = e
Row 20-29	Letter 3 = a	Letter 3 = b	Letter 3 = c	Letter 3 = d	Letter 3 = e
Row 30-39	Letter 4 = a	Letter 4 = b	Letter 4 = c	Letter 4 = d	Letter 4 = e
Row 40-49	Letter 5 = a	Letter 5 = b	Letter 5 = c	Letter 5 = d	Letter 5 = e
Row 50-59	Letter 6 = a	Letter 6 = b	Letter 6 = c	Letter 6 = d	Letter 6 = e

Now someone decoding the message by comparing the altered image against the original can easily determine the message even with this randomness since the letters appear in the message in the same order we find altered bits going down the rows and the decoder knows which groups of columns correspond to each letter. However, with the added randomness, it may be more difficult for someone who lacks the original image to detect something is wrong with the bit patterns of the modified image. Additionally, by spreading the letters out across all the rows of the image, I reduce the chances of steganalysis detecting something wrong with a small portion of the image. This method does have a rather low data capacity (at most the number of rows of the image), but this also increases its strength against steganalysis since there are fewer alterations to be detected.

In addition to the randomness built directly into the data embedding, my program offers options for adding random noise to the image before embedding data into it. The noise is added by selecting random pixels and flipping their LSB before the secret data is even embedded (and then using this altered cover as the new cover). The hope here is to create a cover with a bit pattern that is random in such a way that it is then difficult to tell covers apart from stego-objects, that is, covers that might seem themselves like stego-objects and thus confuse steganalysis tactics.

Taking this added randomness even one step further, I implemented the ability for my program to generate its own random images to be used for covers. A fully random image presents no patterns in its least significant bits, and so it is (nearly) impossible to reliably detect alterations in bit patterns caused by my steganography method since there are no patterns to begin with.

The problem with this is that the complete lack of a pattern in the cover image may itself arouse suspicion of an outside observer, as would the fact that communicators are sending images to one another that are composed of entirely random pixels. As a counter to this, I modified my random image generator to generate

images with pixels that are similar to those surrounding them. The result (an example of which is seen in section 5.2 of "Unseen: An Overview of Steganography and Presentation of an Associated Java Application") is a fairly random image that still displays visual patterns and thus could potentially pass as a form of modern art, which is less suspicious than a fully random image.

The goal of all of this is to generate both covers and stego-objects that are statistically very similar. Some specific results are presented in part in the associated paper ("Unseen"). I found overall that it is very possible to generate cover/stego pairs that are almost identical statistically, thus making it nearly impossible for steganalysis to identify a single image as either cover or stego. Not every random image will have this property, but due to the nature of randomness, as large number are expected to.

Future work on this application could include refining the random image generator, exploring different ways to add randomness to images and to the embedding of messages within images, and altering images in other (perhaps larger) ways, such as flipping or shifting pixels. Though altering pixel locations or groups of pixels may cause visible differences in the image, these will only be noticeable to those who have both the cover and stego-object, and may make it more difficult for a computer viewing only one image to determine whether that image is a cover or a stego-image, as can be seen from statistical analysis of runs of the program presented in "Unseen: An Overview of Steganography and Presentation of an Associated Java Application."

Last Modified: April 20, 2009
This and other papers on latest advances in network security are available on line at http://www.cse.wustl.edu/~jain/cse571-09/index.html
SHARE Back to Raj Jain's Home Page