An Overview of Steganography

GARY C. KESSLER

Gary Kessler Associates, Burlington, Vermont, USA

CHET HOSMER

Allen Corporation, Conway, South Carolina, USA

Abstract

Steganography is the art of *covered*, or *hidden*, *writing*. The purpose of steganography is covert communication—to hide the existence of a message from a third party. Knowledge of steganography is of increasing importance to individuals in the law enforcement, intelligence, and military communities. This chapter provides a high-level introduction to methods and tools for both hiding information (steganography) and detecting hidden information (steganalysis). This chapter is technical, in that it uses many examples using the current tools of the trade, without delving into the deeper mathematics, although references are provided to some of the ongoing research in the field. While this chapter provides a historical context for stego, the emphasis is on digital applications, focusing on hiding information in digital image or audio files. Examples of software tools that employ steganography to hide data inside of other files as well as software to detect such hidden files will also be presented.

ADVANCES IN COMPUTERS, VOL. 83
ISSN: 0065-2458/DOI: 10.1016/B978-0-12-385510-7.00002-3

1. Introduction

Steganography is the art of *covered*, or *hidden*, *writing*. The purpose of steganography is *covert communication* to hide a message from a third party. This differs from *cryptography*, the art of *secret writing*, which is intended to make a message unreadable by a third party but does not necessarily hide the very existence of the secret communication. While steganography is separate and distinct from cryptography, there are many analogies between the two and, in fact, some authors categorize steganography as a form of cryptography, as *hidden* communication certainly is a form of *secret* writing [1]. Nevertheless, this chapter will treat stego as a separate and independent field of study.

Steganography has been—and continues to be—used for the purpose of hiding the fact that two parties are communicating. Aside from an interesting research problem, stego has a number of nefarious applications, however, most notably hiding records of illegal activity, financial fraud, industrial espionage, and communication among members of criminal or terrorist organizations [2]. As such, stego is of interest—either for the purpose of transmitting covert messages or detecting them—to the diplomatic, military, intelligence, criminal, terrorist, and law enforcement communities.

1.1 A Brief History

Although the term *steganography* was only coined at the very end of the fifteenth century, the use of stego dates back several millennia. Soon after humans could write (approximately 2000 BC), we learned how to write in secret codes and, soon thereafter, how to communicate covertly. In ancient times, messages were hidden on the back of wax-writing tables, written on the stomachs of rabbits, or tattooed on the scalp of slaves. Invisible ink has been in use for centuries—for fun by kids and students, for serious espionage by spies and terrorists. Microdots and microfilm, a staple of war and spy movies, came about after the invention of photography [3–6].

Steganography hides the covert message but not necessarily the fact that two parties are communicating with each other. As will be discussed below, messages can be hidden in an e-mail message, a photograph posted on a Facebook page, or a drawing given to a friend. While there are many high-technology methods of hiding information, low-tech methods may work just as well for a period of time.

1.2 Terms, Concepts, and Classifications of Steganography

The stego process generally involves placing a *hidden message* within some transport medium, called the *carrier*. The hidden message is embedded within the carrier to form the *stego medium*. The use of a *stego key* may be employed for encryption of the hidden message and/or as a randomization seed for the stego algorithm. In summary:

$$\text{stego\_medium} = \text{hidden\_message} + \text{carrier} + \text{stego\_key}$$

Figure 1 shows a common taxonomy of steganographic techniques [1,3]:

- *Technical steganography* uses scientific methods to hide a message, such as the use of invisible ink or microdots and other size reduction methods. This chapter will not address technical steganography methods.

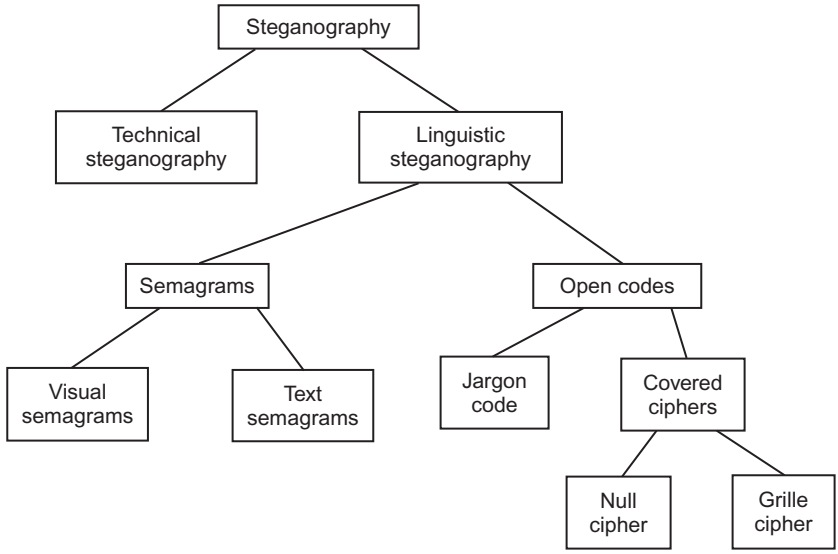

FIG. 1. Classification of steganography techniques (adapted from [1]).

- *Linguistic steganography* hides the message within the carrier in some nonobvious ways and is further categorized as *semagrams* or *open codes*.
- *Semagrams* hide information by the use of symbols or signs. A *visual semagram* uses innocent-looking or everyday physical objects to convey a message, such as doodles, the positioning of items on a desk or Web site, or the placement of a flag on a balcony (à la Deep Throat of Watergate fame). A *text semagram* hides a message by modifying the appearance of the carrier text, such as subtle changes in font size or type, adding extra spaces, or different flourishes in letters or handwritten text.
- *Open codes* hide a message within a legitimate carrier message in ways that are not obvious to an unsuspecting observer. The carrier message is sometimes called the *overt communication*, while the hidden message is the *covert communication*. This category is subdivided into *jargon codes* and *covered ciphers*.
- *Jargon code*, as the name suggests, uses language that is understood by a group of people but is meaningless to others. Jargon codes include warchalking (symbols used to indicate the presence and type of wireless network signal [7]), underground terminology, or an innocent conversation that convey special meaning because of facts known only to the speakers. A subset of jargon codes is *cue codes*, where certain prearranged phrases convey meaning. Jargon codes are not addressed further in this chapter.

- *Covered, or concealment, ciphers* hide a message openly in the carrier medium so that it can be recovered by anyone who knows the secret for how it was concealed. A *grille cipher* employs a template that is used to cover the carrier message; the words that appear in the openings of the template are the hidden message. A *null cipher* hides the message according to some prearranged set of rules, such as "read every fifth word" or "look at the third character in every word."

As an increasing amount of data is stored on computers and transmitted over networks, it is no surprise that steganography has entered the digital age. On computers and networks, stego applications allow for someone to hide any type of binary file into many other types of binary files, although image and audio files are today's most common carriers.

1.3 Steganography Versus Digital Watermarking

Steganography provides some very useful and commercially important functions in the digital world, most notably *digital watermarking*. In this application, an author can embed a hidden message in a file so that he/she can later assert their ownership of intellectual property and/or ensure the integrity of the content. An artist, for example, could post some original artwork on a Web site. If someone else should "steal" the file and claims the work as his/her own, the artist can later prove ownership because only he/she can recover the watermark [3,8,9]. While conceptually similar to stego, digital watermarking usually has different technical goals, namely:

- Generally, the watermark is a small amount of repetitive information that is inserted into the carrier,
- It is not always necessary to hide the watermarking information from the viewer, and
- It is useful if the watermark can be removed while maintaining the integrity of the carrier file.

Regardless of how one looks at it, however, watermarking is not the same as steganography; the two methods use different algorithms, have different purposes, and provide different levels of threat [10].

1.4 "Time-Sensitive" Steganography

As a slight aside to the art and science of steganography, it is worth noting that no covert channel needs to remain covert forever. While some academics eschew the thought of using any stego method that has not been shown to be immune from many

types of attack, most practitioners recognize that most secret messages have a finite lifetime. For this reason, a stego method is sufficient for its task if it can hold the secret long enough to suit the aims of the covert communicators. Stated another way, even a poor stego method may be of use if it can hold a secret longer than it takes an adversary to detect and decode the covert channel.

Time-sensitive steganography refers to the criteria in the design or selection of a stego method. A practitioner needs a stego scheme that has usability and implementation criteria that meet the following:

$$T_{CRITICAL} < T_{DETECT} + T_{DECODE}$$

As stated above, a stego method needs to be strong enough so that the time to detect the covert channel (T_{DETECT}) plus the amount of time to decode the covert channel (T_{DECODE}) is greater than the required lifetime of the channel ($T_{CRITICAL}$). $T_{CRITICAL}$ is up to the user; in some cases, one only needs to keep the secret for a few days or weeks; in other cases, one would want the covert channel to be kept secret for years or decades.

2. Low-Tech Stego Methods

A variety of low-technology stego methods are presented in this section. While some of the methods described here employ computers in one way or another, they all require some form of human intelligence to prepare and interpret. Low-tech methods, therefore, are often difficult to detect using only automated methods and, indeed, may be improvisational. This section will discuss various forms of semagrams, concealment ciphers, and other methods.

2.1 Semagrams

Semagrams hide information using special symbols or signs. These signs can be in the form of pictures or objects (visual semagram), or subtlety altered written documents (text semagrams). A visual semagram conveys a message using items that can be seen, such as articles of clothing or the placement of items in a room. Pictures and images on a Web site can also convey information, such as the position of items on a particular Web page or the order of pages at a site. The orientation of photographs on a Web page might also convey meaning (Fig. 2). Of course, any such arrangement of items requires an *a priori* agreement between the communicating parties.

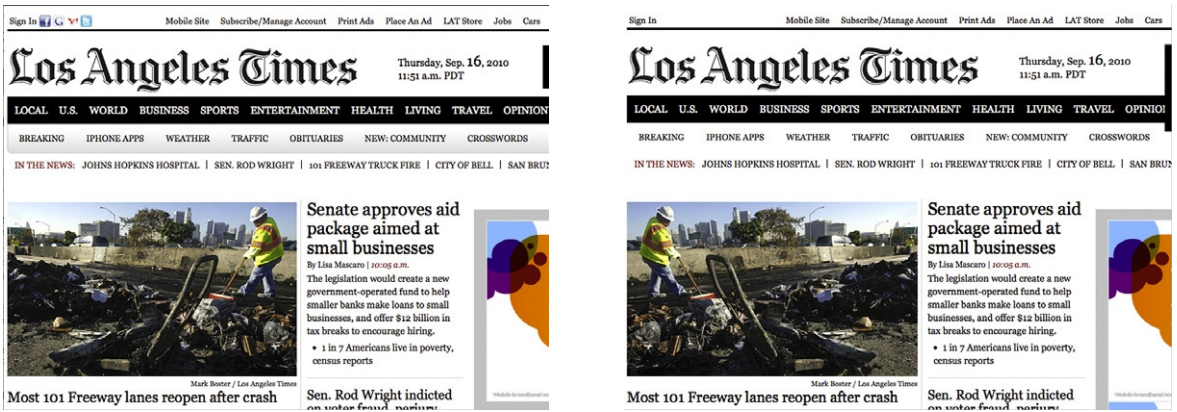

FIG. 2. Screen shots of LATimes.com Web site. The real screen shot is on the left and the altered site is on the right.

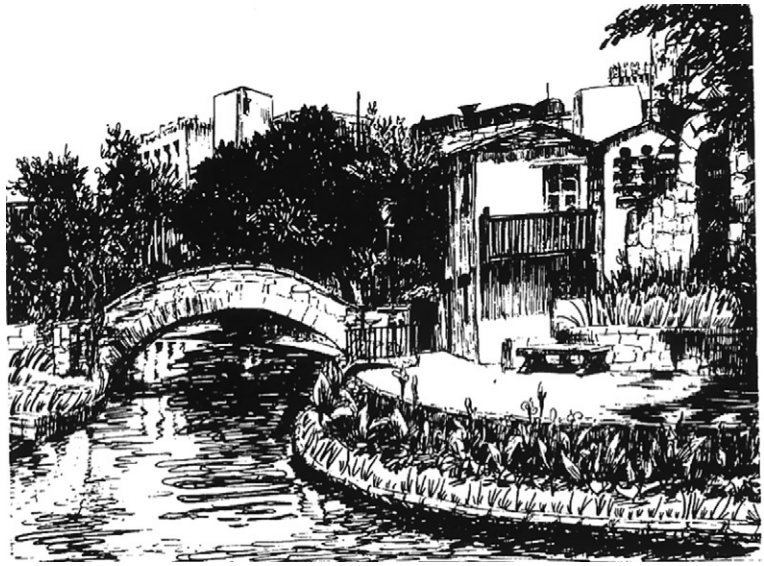

FIG. 3. Drawing of the San Antonio River.

Another form of visual semagram is to hide a code within a picture. Figure 3 shows a well-known drawing of the San Antonio River [1]. This drawing was created by a group of government censors in San Antonio to commemorate the visit of their commander, Col. Harold Shaw, in May 1945.

The reader should pay particular attention to the grass along the river; each clump of grass represents a single Morse code character, with the short blades representing a dot and a long blade representing a dash. The complete message says:

> Compliments of CPSA MA to our chief Col Harold R Shaw on his visit to San Antonio May 11th 1945

2.2 Concealment Ciphers

Concealment ciphers are an old method of hiding messages. These type of ciphers hide a message openly in the carrier medium so that it can be recovered by anyone who knows the secret for how it was concealed. A grille cipher, for example, employs a template that reveals the message once applied to the original carrier. One well-known example is that of a letter written by British Lt. General Sir Henry Clinton to General John Burgoyne in August 1777 (Fig. 4).

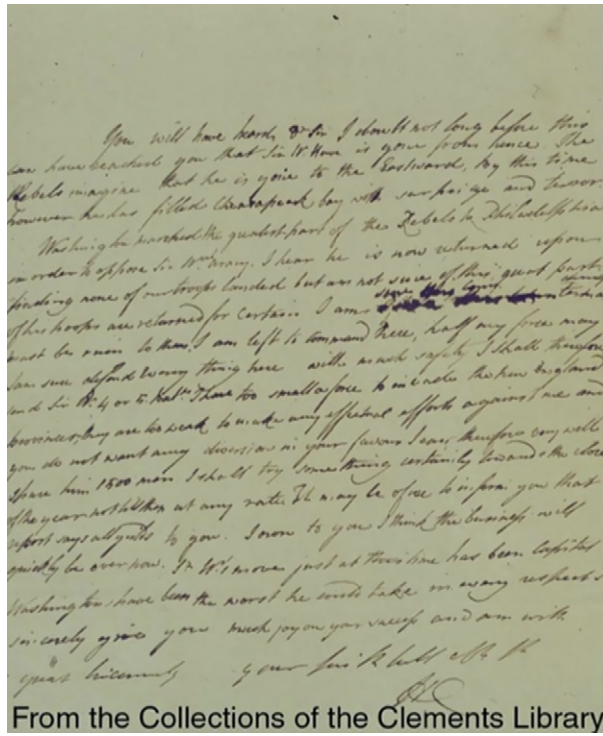

FIG. 4. Original letter written by Henry Clinton to John Burgoyne [11].

The text of the letter reads [11]:

You will have heard, Dr Sir I doubt not long before this
can have reached you that Sir W. Howe is gone from hence. The
Rebels imagine that he is gone to the Eastward. By this time
however he has filled Chesapeak bay with surprize and terror.
Washington marched the greater part of the Rebels to Philadelphia
in order to oppose Sir Wm's. army. I hear he is now returned upon
finding none of our troops landed but am not sure of this, great part
of his troops are returned for certain. I am sure this countermarching
must be ruin to them. I am left to command here, half of my force may
I am sure defend everything here with much safety. I shall therefore
send Sir W. 4 or 5 Bat [talio] ns. I have too small a force to invade the New England
provinces; they are too weak to make any effectual efforts against me and
you do not want any diversion in your favour. I can, therefore very well

spare him 1500 men. I shall try some thing certainly towards the close of the year, not till then at any rate. It may be of use to inform you that report says all yields to you. I own to you that I think the business will quickly be over now. Sr. W's move just at this time has been capital. Wahingtons have been the worst he could take in every respect. sincerely give you much joy on your success and am with great Sincerity your [ ]
HC

Clinton's letter was intended to be read by Burgoyne using a grille (Fig. 5). Once the grille is applied, the letter reads [11]:

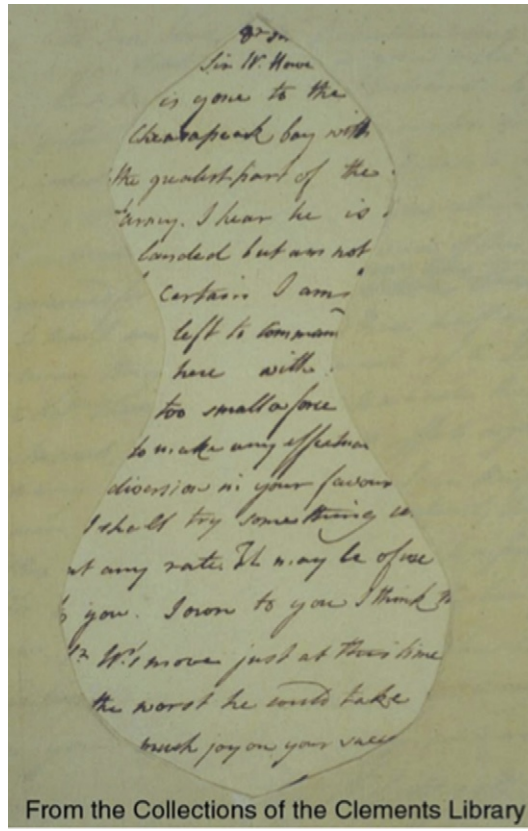

FIG. 5. Grille applied to letter written by Henry Clinton to John Burgoyne [11].

Sir. W. Howe
is gone to the
Chesapeak bay with
the greatest part of the
army. I hear he is
landed but am not
certain. I am
left to command
here with
too small a force
to make any effectual
diversion in your favour.
I shall try something
at any rate. It may be of use
to you. I own to you I think
Sr W's move just at this time
the worst he could take.
Much joy on your success.

A null cipher also hides a message openly in the carrier text according to some prearranged agreement. One of the simplest null ciphers is shown in the classic examples below:

PRESIDENT'S EMBARGO RULING SHOULD HAVE IMMEDIATE NOTICE. GRAVE SITUATION AFFECTING INTERNATIONAL LAW. STATEMENT FORESHADOWS RUIN OF MANY NEUTRALS. YELLOW JOURNALS UNIFYING NATIONAL EXCITEMENT IMMENSELY.

APPARENTLY NEUTRAL'S PROTEST IS THOROUGHLY DISCOUNTED AND IGNORED. ISMAN HARD HIT. BLOCKADE ISSUE AFFECTS PRETEXT FOR EMBARGO ON BYPRODUCTS, EJECTING SUETS AND VEGETABLE OILS.

The German Embassy in Washington, DC sent these messages in telegrams to their headquarters in Berlin during World War I [5]. Reading the first character of every word in the first message or the second character of every word in the second message will yield the following hidden text, referring to US General John Pershing:

PERSHING SAILS FROM N.Y. JUNE 1

2.3 Other Methods

It is important to note that one does not need any special tools or skills to hide messages in digital files using other variances of these low-tech methods. An image or text block can be hidden under another image in a PowerPoint file, for example.

Messages can be hidden in the properties of a Word file. Messages can be hidden in comments within Web pages or in other formatting vagaries that are ignored by browsers [12]. Text can be hidden as line art in a document by putting the text in the same color as the background and placing another drawing in the foreground; the recipient could retrieve the hidden text by changing its color (J. Seward, personal communication, January 2004). These are all decidedly low-tech mechanisms—but can be very effective.

3. Digital Technology Basics

Most of today's most commonly used digital steganography techniques employ graphical images or audio files as the carrier medium. It is instructive, then, to review image and audio encoding before discussing how steganography and steganalysis work with these carriers.

3.1 Digital Images and Color

The Red–Green–Blue (RGB) color cube (Fig. 6) is a common means with which to represent a given color by indicating the relative intensity of its three component colors—red, green, and blue—each with their own axis. The absence of all colors

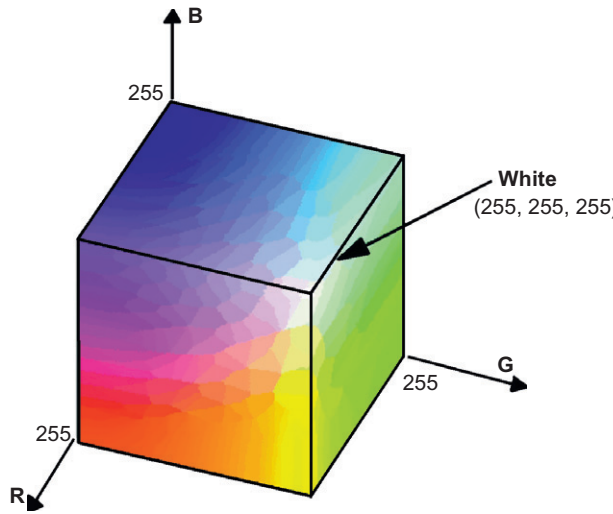

FIG. 6. The RGB color cube [13].

yields black, shown as the intersection of the zero point of the three-color axes (not visible in the figure). The mixture of 100% red, 100% blue, and the absence of green form magenta (255, 0, 255); cyan is 100% green and 100% blue without any red (0, 255, 255); and 100% green and 100% red with no blue combine to form yellow (255, 255, 0). White is the presence of all three colors (255, 255, 255).

Figure 7 shows the RGB intensity levels of some given color. Each RGB component is specified by a single byte, so that the values for each color intensity can vary from 0 to 255. This particular shade is denoted by a red level of 191 (hex 0xBF), a green level of 29 (hex 0x1D), and a blue level of 152 (hex 0x98). One picture element (pixel, or pix) of the color shown in the "new" block in the dialog box in the figure, then, would be encoded using 24 bits as 0xBF1D98. This 24-bit encoding scheme supports 16,777,216 (2^{24}) unique colors [14,15].

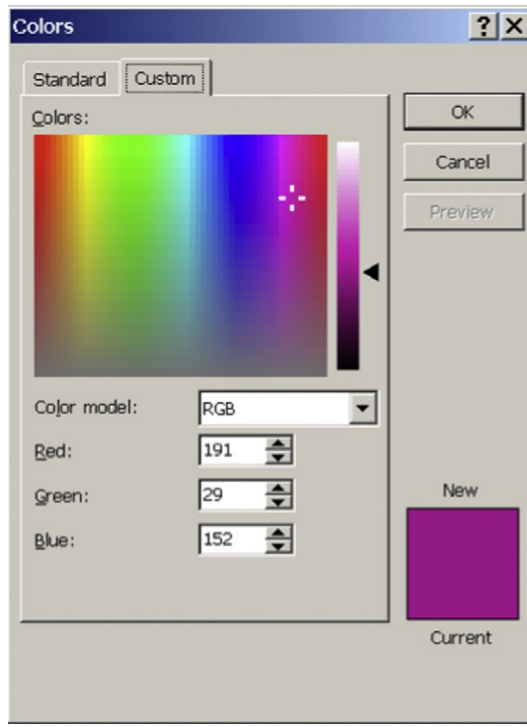

FIG. 7. This color selection dialog box shows the red, green, and blue (RGB) levels of this particular color.

Most digital image applications today support 24-bit True Color, where each pixel is encoded in 24 bits, comprising the three RGB bytes described above. Other applications encode color using 8 bits/pix. These schemes also use 24-bit true color but employ a palette that specifies which colors are used in this image file. Each pix is encoded in 8 bits, where the value points to a 24-bit color entry in the palette. This method limits the unique number of colors in a given image to 256 (2^8). The choice of color encoding obviously affects image size. A 640×480 pixel image using 8-bit color would occupy approximately 307 KB ($640 \times 480 = 307{,}200$ bytes), while a 1400×1050 pix image using 24-bit true color would require 4.4 MB ($1400 \times 1050 \times 3 = 4{,}410{,}000$ bytes).

Color palettes and 8-bit color are commonly used with Graphics Interchange Format (GIF) and Bitmap (BMP) image formats. GIF and BMP are generally considered to offer *lossless compression* because the image recovered after encoding and compression is bit-for-bit identical to the original image [15].

The Joint Photographic Experts Group (JPEG) image format uses discrete cosine transforms (DCTs) rather than a pix-by-pix encoding. In JPEG, the image is divided into 8×8 blocks for each separate color component. The goal is to find blocks where the amount of change in the pixel values (the *energy*) is low. If the energy level is too high, the block is further subdivided into 8×8 subblocks until the energy is low enough. Each 8×8 block (or subblock) is transformed into 64 DCT coefficients that approximate the luminance (brightness, darkness, and contrast) and chrominance (color) of that portion of the image. JPEG is generally considered to be *lossy compression* because the image recovered from the compressed JPEG file is a close approximation of, but not identical to, the original [15–17].

3.2 Digital Audio

Audio encoding involves converting an analog signal to a bit stream. Analog sound—voice and music—is represented by sine waves of different frequencies. The human ear can hear frequencies nominally in the range of 20–20,000 cycles/second (Hertz, or Hz). Sound is analog, meaning that it is a continuous signal. Storing the sound digitally requires that the continuous sound wave be converted to a set of samples that can be represented by a sequence of zeroes and ones [18].

Analog-to-digital conversion is accomplished by sampling the analog signal (with a microphone or other audio detector) and converting those samples to voltage levels. The voltage, or signal, level is then converted to a numeric value using a scheme called pulse code modulation (PCM). The device that performs this conversion is called a *coder–decoder*, or *codec* [18].

PCM provides only an approximation of the original analog signal (Fig. 8). If the analog sound level, for example, is measured at a 4.86 level, it would be converted to

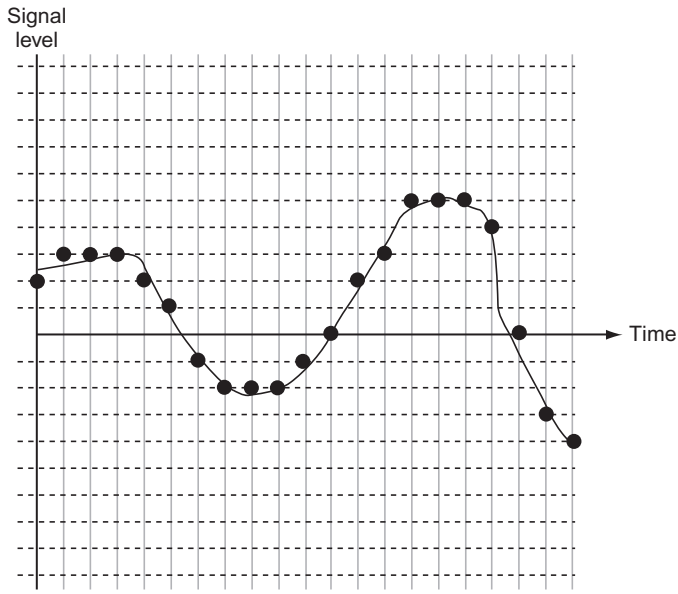

FIG. 8. Simple pulse code modulation (PCM).

a 5 in PCM; this is called *quantization error*. Different audio applications define a different number of PCM levels so that this "error" is nearly undetectable by the human ear. The telephone network converts each voice sample to an 8-bit value (255 PCM levels), while music applications generally use 16-bit values (65,535 PCM levels) [18,19].

Analog signals need to be sampled at a rate of twice the highest frequency component of the signal so that the original can be correctly reproduced from the samples alone. In the telephone network, the human voice is carried in a frequency band 0–4000 Hz (although only about 400–3400 Hz is actually used to carry voice); therefore, voice is sampled 8000 times per second (an 8 kHz sampling rate). Music audio applications assume the full spectrum of the human ear and generally use a 44.1 kHz sampling rate [18,19].

The bit rate of uncompressed music can be easily calculated from the sampling rate (44.1 kHz), PCM resolution (16 bits), and number of sound channels (2) to be 1,411,200 bits per second. This would suggest that a 1-minute audio file (uncompressed) would occupy 10.6 MB ($1{,}411{,}200 \times 60/8 = 10{,}584{,}000$). Audio files are, in fact, made smaller by using a variety of compression techniques. One obvious method is to reduce the number of channels to 1 or to reduce the sampling rate, in

TABLE I
COMMON DIGITAL AUDIO FORMATS (FROM [19])

Audio type	File extension	Codec
AIFF (Mac)	.aif, .aiff	PCM (or other)
AU (Sun/Next)	.au	μ-law (or other)
CD audio (CDDA)	n/a	PCM
MP3	.mp3	MPEG Audio Layer III
Windows Media Audio	.wma	Microsoft proprietary
QuickTime	.qt	Apple Computer proprietary
RealAudio	.ra, .ram	Real Networks proprietary
WAV	.wav	PCM (or other)

some cases as low as 11 kHz. Other codecs use proprietary compression schemes. All these solutions reduce the quality of the sound. Table I lists some of the common digital audio formats.

3.3 Payload Compression and Steganography

There are two fundamental reasons for steganography algorithms compressing payloads prior to embedding, namely, to reduce the size of the payload or to create a more randomized payload.

By reducing the size of the payload through compression, the overall impact on the carrier file is also reduced (Fig. 9). This is quite obvious, although not all steganography programs provide this capability automatically. For example, the F5 algorithm—one of the best JPEG embedding algorithms, developed by Andreas Westfeld of Technische Universität Dresden—leaves *a priori* payload compression up to the user. Since compression of documents, spreadsheets, and other noncompressed files can significantly reduce the size of the payload, the ability to embed more information safely (without detection) improves as the payload size is reduced. For some steganography applications, such as S-Tools, the program provides both information regarding the maximum size of the payload and the ability to control the compression method.

The second and less obvious reason for applying compression is to create a more randomized payload (Fig. 10). This is particularly important when embedding text or simple messages into a carrier file. Clearly, encryption can be used for the same purpose. The underlying rationale behind the randomization of the payload is to ensure the 50% rule; since most steganography makes slight alterations to the least significant bit (LSB; RGB values, Palette Colors, JPEG DCT coefficients, etc.),

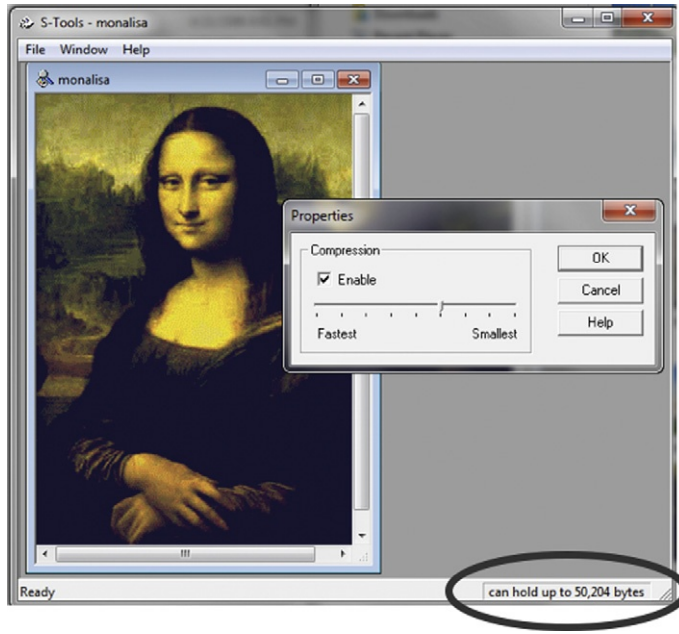

FIG. 9. S-Tools showing the potential payload capacity of this carrier file.

there is a 50/50 chance that the bit you are modifying will already be in the correct state if the data being embedded is random. In this case, there is a 50% chance that no actual alteration would be made, making the carrier an even better approximation of the original, unmodified carrier.

4. Steganography and Digital Carrier Files

This section will discuss the major classifications of steganography methods that operate on digital carriers. Image and audio files are the easiest and most common carrier media because of the plethora of potential carrier files already in existence, the ability to create an infinite number of new carrier files, and the easy access to stego software that will operate on these carriers. For that reason, most of the examples below operate on image and audio files. That said, digital carriers also include network traffic, so methods operating over Transmission Control Protocol/Internet Protocol (TCP/IP) packets will also be addressed, including stego in digital voice.

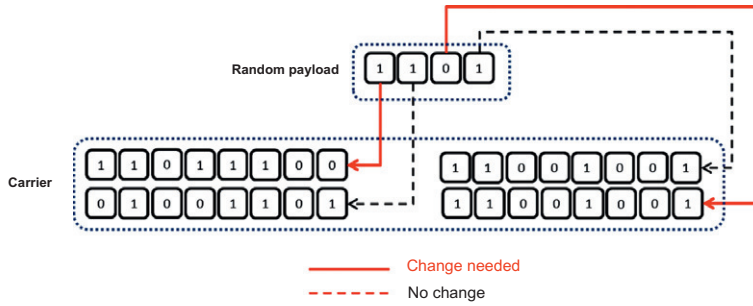

FIG. 10. A compressed payload being used to add randomization to the carrier.

4.1 Least-Significant Bit Overwriting

LSB overwriting (or substitution) is a common stego method when using audio and image carrier files. The term LSB comes from the numeric significance of the bits in a byte. The high-order, or most significant, bit is the one with the largest arithmetic value (i.e., $2^7 = 128$), while the low-order, or least significant, bit is the one with the smallest arithmetic value (i.e., $2^0 = 1$).

As a simple example of LSB substitution, imagine hiding the character "G" across the following eight bytes of a carrier file (the LSBs are underlined):

10010101 00001101 11001001 10010110
00001111 11001011 10011111 00010000

A "G" is represented in the American Standard Code for Information Interchange (ASCII)/International Alphabet 5 (IA5) as the binary string 01000111. These 8 bits can be written to the LSB of each of the 8 carrier bytes as follows:

10010100 00001101 11001000 10010110
00001110 11001011 10011111 00010001

In the sample above, note that only half of the LSBs were actually changed (shown above in *italics*). This actually makes some sense since there should only be about a 50% mismatch when substituting one set of zeroes and ones with another set of zeroes and ones.

LSB substitution can be used to overwrite legitimate RGB color encodings or palette pointers in GIF and BMP files, coefficients in JPEG files, and PCM levels in audio files. By overwriting the LSB, the numeric value of the byte changes very little and is least likely to be detected by the human eye or ear. (Imagine how unlikely the reader would be to detect a red level change from 191 to 190, a green level change from 29 to 28, and/or a blue level change from 152 to 153 in the color in Fig. 7.)

LSB substitution is a simple, albeit common, technique for steganography. Its use, however, is not necessarily as simplistic as the method sounds. Only the most naive stego software would merely overwrite every LSB with hidden data; almost all use some sort of means to randomize the actual bits in the carrier file that are modified. This is one of the factors that makes stego detection so difficult.

One example of hiding information in a digital carrier uses a program called S-Tools. In this example, an 11,067-byte GIF image of the Burlington, Vermont airport (Fig. 11) will be hidden in a WAV audio file.

S-Tools is a program, written by Andy Brown, which can hide information inside GIF, BMP, and WAV files. S-Tools uses LSB substitution in files that employ lossless compression, such as 8- or 24-bit color image and PCM audio files. S-Tools employs a password for LSB randomization and can encrypt data using the Data Encryption Standard (DES), International Data Encryption Algorithm (IDEA), Message Digest Cipher (MDC), or Triple-DES [6,15,20,21]. Figure 12 shows a signal level comparison between a WAV carrier file before (left) and after (right) the airport map was hidden. The original WAV file is 178,544 bytes in length, while the stego WAV file is 178,298 bytes in length. Although the relatively small size of the figure makes it hard to see details, some differences are noticeable at the beginning and end of the audio sample; that is, during periods of silence. (Some stego tools have built-in intelligence to avoid the low-intensity portions of the signal.) Audio files are well suited to information hiding because they are usually relatively large, making it relatively easy to hide, but difficult to find, small volumes of hidden data.

FIG. 11. A GIF image file of the map of the Burlington, VT airport.

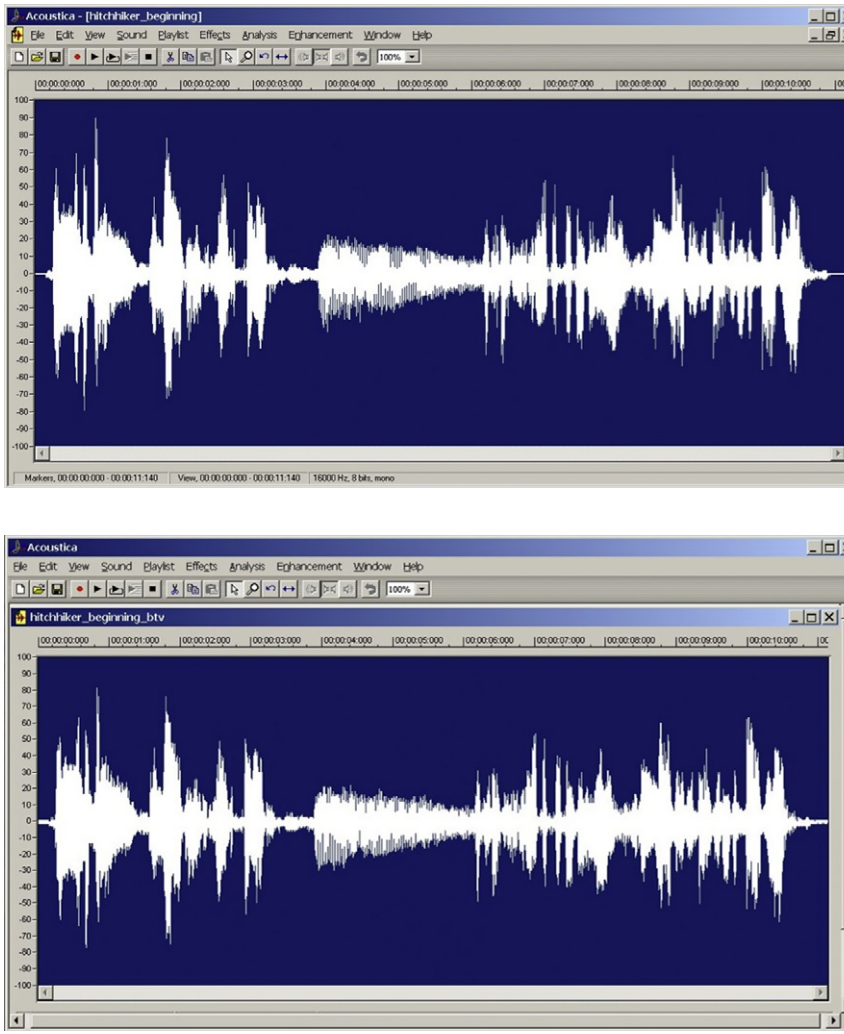

FIG. 12. The signal level comparisons between a WAV carrier file before (above) and after (below) the airport map is inserted.

4.2 Encoding Algorithm Modification

JP Hide-&-Seek (JPHS) by Allan Latham is designed to be used with JPEG files and lossy compression. JPHS alters the DCT coefficients used by the JPEG algorithm. The Blowfish crypto algorithm is used for LSB randomization and encryption [20,21]. Figure 13 shows an example of JPEG file with the airport map embedded in it. The original carrier file is 207,244 bytes in size and contains 224,274 unique colors; the stego file is 207,275 bytes in size and contains 227,870 unique colors. There is no color palette to look at because JPEG uses 24-bit color coding and DCT.

4.3 Grammar Selection

Another way in which to hide a message in plain sight employs grammar-based mimicry [6]. In this scheme, the communicating parties have a prearranged set of phrases and text clauses that convey meaning. The bit string that represents the hidden message dictates the phrases and clauses that will appear in the carrier text.

An example of grammar-based stego hides a message in a carrier that otherwise looks like spam, such as the following:

> Dear Friend, This letter was specially selected to be sent to you! We will comply with all removal requests! This mail is being sent in compliance with Senate bill

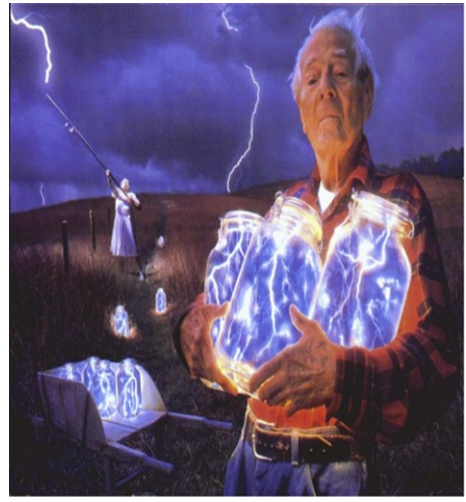

FIG. 13. A JPEG carrier file containing the airport map.

1621; Title 5; Section 303! Do NOT confuse us with Internet scam artists. Why work for somebody else when you can become rich within 38 days! Have you ever noticed the baby boomers are more demanding than their parents & more people than ever are surfing the web! Well, now is your chance to capitalize on this! WE will help YOU sell more & SELL MORE. You can begin at absolutely no cost to you! But don't believe us! Ms Anderson who resides in Missouri tried us and says "My only problem now is where to park all my cars". This offer is 100% legal. You will blame yourself forever if you don't order now! Sign up a friend and your friend will be rich too. Cheers! Dear Salaryman, Especially for you - this amazing news. If you are not interested in our publications and wish to be removed from our lists, simply do NOT respond and ignore this mail! This mail is being sent in compliance with Senate bill 2116, Title 3; Section 306! This is a ligitimate business proposal! Why work for somebody else when you can become rich within 68 months! Have you ever noticed more people than ever are surfing the web and nobody is getting any younger! Well, now is your chance to capitalize on this. We will help you decrease perceived waiting time by 180% and SELL MORE. The best thing about our system is that it is absolutely risk free for you! But don't believe us! Mrs Ames of Alabama tried us and says "My only problem now is where to park all my cars". We are licensed to operate in all states! You will blame yourself forever if you don't order now! Sign up a friend and you'll get a discount of 20%! Thanks! Dear Salaryman, Your email address has been submitted to us indicating your interest in our briefing! If you no longer wish to receive our publications simply reply with a Subject: of "REMOVE" and you will immediately be removed from our mailing list. This mail is being sent in compliance with Senate bill 1618, Title 6, Section 307. THIS IS NOT A GET RICH SCHEME. Why work for somebody else when you can become rich within 17 DAYS! Have you ever noticed more people than ever are surfing the web and more people than ever are surfing the web! Well, now is your chance to capitalize on this! WE will help YOU turn your business into an E-BUSINESS and deliver goods right to the customer's doorstep! You are guaranteed to succeed because we take all the risk! But don't believe us. Ms Simpson of Wyoming tried us and says "Now I'm rich, Rich, RICH"! We assure you that we operate within all applicable laws. We implore you - act now! Sign up a friend and you'll get a discount of 50%. Thank-you for your serious consideration of our offer.

This message looks like the spam that most Internet users receive every day, which are generally ignored and discarded. This message was created at *spammimic.com*, a Web site that converts a short text message into a text block [22]. The reader will learn nothing by looking at the word spacing or misspellings in the message; the zeroes and ones are encoded by the very choice of the words.

The hidden message in the spam carrier above is:

Meet at Main and Willard at 8:30

The algorithm from *spammimic.com* takes the message above, deciphers it into a bit stream, and then selects phrases from a database based upon the combination and order of bits in the stream. This is the essence of grammar-based steganography.

Text files are not the only type of carrier on which a grammar-selection type of method can be employed; other file types also have characteristics that can be exploited for information hiding. Hydan, for example, can conceal text messages in OpenBSD, FreeBSD, NetBSD, Red Hat Linux, and Windows executable files. Developed by Rakan El-Khalil, Hydan takes advantage of redundancies in the i386 instruction set and inserts hidden information by defining sets of functionally equivalent instructions, conceptually like a grammar-based mimicry (e.g., where ADD instructions are a zero bit, and SUB instructions are a one bit). The program can hide approximately one message byte in every 110 instruction bytes and maintains the original size of the application file. Blowfish encryption can also be employed [23,24].

4.4 Data Appending

Data appending is an almost obvious way of hiding data in a digital carrier. In this method, the data to be hidden is merely piggybacked to the carrier file. This method actually will only work with files that have both a file header (signature) and trailer, because these will demarcate the beginning and, more importantly, end of the carrier.

Figure 14 shows two JPEG files; the original carrier file on the left (61,289 bytes in length) and the carrier file with hidden data on the right (73,211 bytes). The two files look identical, and from the perspective of any JPEG image viewer application, they are.

FIG. 14. A JPEG original (left) and carrier (right) file containing the airport map.

The hidden data was inserted into this JPEG carrier file using a program called Camouflage. Camouflage works by encrypting the data file to be hidden in the carrier (which adds randomization) and then appending it to the original carrier file's data.

The example shown here employs a JPEG file as the carrier. The first 4 bytes of a JPEG file are the file signature (header), comprising the byte pattern `FF D8 FF E0` (ÿØÿà) [the byte pattern `4A 46 49 46 (JFIF)` can also be found starting at byte offset 6]. JPEG files also have a two-byte trailer with the byte pattern `FF D9` (ÿÙ) [25]. A hex view of the portion of the carrier file with the JPEG trailer (at bytes 0xEF67-EF68) clearly shows data after the end of the JPEG image (Fig. 15). Note also that the size of the carrier file with hidden text is roughly the sum of the original carrier file and that of the file containing the image to be hidden.

Inch [26] describes a low-tech method of information hiding using data appending and a file compression application such as RAR or ZIP. Consider an example where one wishes to use a JPEG file as a carrier and wants to hide one or more other files. First, create a ZIP archive of the files that are to be hidden. Then, using the DOS copy command with the /b (binary) switch, append the ZIP file to the JPEG carrier, as follows:

```
copy /b original.jpg+hidden.zip stego.jpg
```

The size of the resultant JPEG file with the hidden data (stego.jpg) will be approximately the sum of the size of the original carrier file (original.jpg) and the

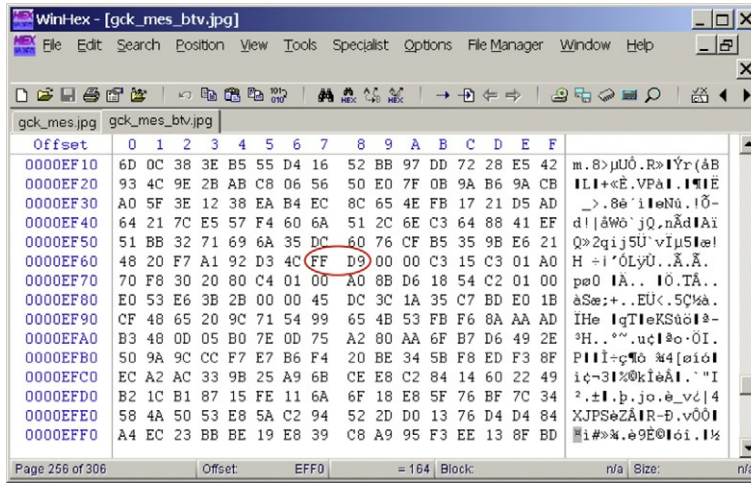

FIG. 15. A hex dump of a portion of the JPEG carrier file, showing the JPEG trailer.

ZIP archive file (hidden.zip). This method is simple and works for two reasons. First, the file has a JPEG extension and will, therefore, be associated with an image viewer. Since JPEG has a file trailer, the image viewer will stop displaying when it finds the trailer. Second, one can open the stego.jpg file directly from the ZIP program. Since this application starts to process the archive from the end of the file, it will find the archived information and not be confused by the presence of anything before the ZIP archive header. This method is not perfect and will not stand against analysis for very long, but it can work for a sufficient amount of time for the two parties.

4.5 Color Palette Modification

One other way to hide information in a paletted image is to alter the order of the colors in the palette or use LSB encoding on the palette colors rather than on the image data. These methods are potentially weak, however. Many graphics software tools order the palette colors by frequency, luminance, or other parameter, and a randomly ordered palette stands out under statistical analysis [27].

Gif-It-Up is a Nelsonsoft program that hides information in GIF files using color palette modification (and includes an encryption option). Figure 16 shows a GIF image of the Washington, DC mall at night where Gif-It-Up has been used to insert the Burlington airport map image from above. The original carrier file is 632,778 bytes in length and uses 249 unique colors, while the stego file is 677,733 bytes in

FIG. 16. A GIF carrier file containing the airport map.

length and uses 256 unique colors. The file size is larger in the stego file because of a color extension option used to minimize distortion in the stego image; if color extension is not employed, the file size differences are slightly less noticeable.

Figure 17 shows the carrier file's palettes before and after message insertion. Like all palette modification programs that act on 8-bit color images, Gif-It-Up modifies the color palette and generally ends up with many duplicate color pairs. Note that the original palette (on the left in the figure) appears to be slightly more ordered in terms of color organization than the palette from the stegoed image; this is one of the results of palette modification.

Color palette modification is one area of steganalysis that is sometimes overlooked, largely due to the general opinion that these palette-based carrier files have minimal payload carrying capabilities. This is caused first by the nature of palette images and second due to the compression that is applied to GIF and other palette types. In addition, the sophistication of algorithms that have been traditionally created for palette image types is far less than those applied to JPEG images, for example.

Figure 18 shows another view of a carrier file before and after data is hidden, using the program Gif-It-Up, where the image on the left is the original (unmodified) file and the image on the right is the one with stego data. (Note in this case that the process started with a GIF image to produce a new GIF image, whereas a more effective method would be to start with a True Color image to produce a GIF from

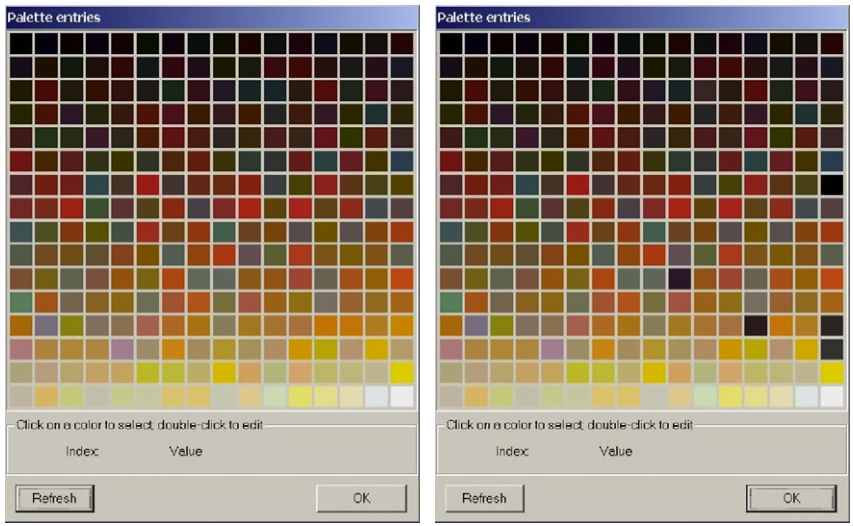

FIG. 17. The palette from the Washington mall carrier file before (left) and after (right) the map file was hidden.

FIG. 18. Palette-based steganography; original carrier of the left, stego carrier on the right.

the True Color image while embedding the payload during the conversion. As it happens, very few, if any, stego programs offer this option for palette images).

Examining the basic properties of each image, we see the two files differ in one parameter, namely *file size*. The reason for this difference, however, may not be the expected one. The most common approach used to perform steganography on palette images is to create close color pairs, or *color buddies*, that can be used to encode a binary value into the pixel data. This common technique makes alterations (not additions) to the palette and alters the pixel data values to correspond with the new palette. Once again, however, this does not increase the number of pixel entries.

If the palette does not change in size and the number of pixel entries remains the same, why then does the file size change? The answer is compression. GIF images are compressed using the lossless Lempel–Ziv–Welch (LZW) compression [28]. As the pixel data has changed based on stego-based palette modification, the compression can be either less or more efficient and thus impacts the resulting file size.

GIF and other palette-based images are generally lower quality images due to the reduced color availability. Common palette images contain a fixed palette of 256

colors (extended palettes are available in some formats). While each of paletted color can be any one from the 24-bit RGB array (~16.8 million), a single image can only contain 256 unique colors. For this reason, any rendering of the image will have reduced color resolution. For this reason, GIF images are grainier than other image types. When stego is applied, the graininess is exaggerated. As shown in Fig. 19, graininess is present in both the original (left) and stego (right) carrier images but is much more predominant in the stego image (making the image appears pixelated). This is caused by the further reduction in colors that takes place through the stego process that replaces unique colors in the palette with close pairs.

These palette-based anomalies can be easily detected through algorithmic analysis of the palette. The simplified concept is to examine the distance between colors in the palette; as only 256 colors are available, the standard encoding algorithms will not naturally select close colors since humans cannot distinguish close color pairs. The encoding algorithms instead select a wider range of colors to populate the palette. When stego is applied and the palette is modified using the close color pair process, the close pairs can be easily identified. The first trick in this visual identification is to sort the palette by color (as the palette is not normally ordered this way), which allows rapid identification of the distribution of colors present in the palette. This is shown in Fig. 20, with the original palette on the left and the stego image palette on the right. The greater diversity of colors that exist in the original versus the stego modified version is obvious.

Close color pairs can be further illustrated by examining the binary values of adjacent palette color entries in the stego image (Fig. 21). Note in this example that the only difference between these two palette entries is the least significant bit of the red color value (the first byte of the 24-bit RBG value). This is anomalous to any known palette encoding method and is attributed directly to palette modification for data hiding.

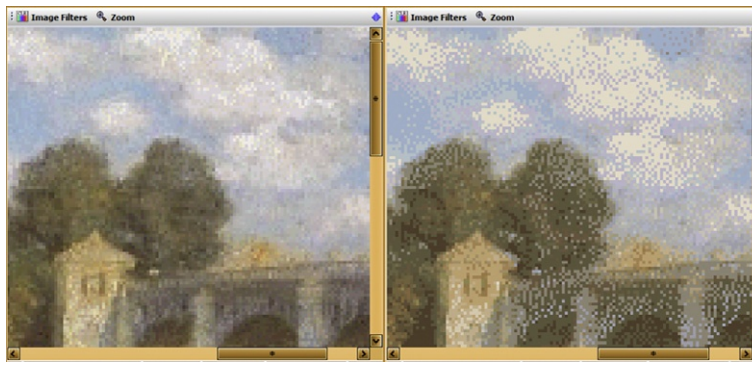

FIG. 19. GIF image enlargement reveals distorsion in stego carrier image (right).

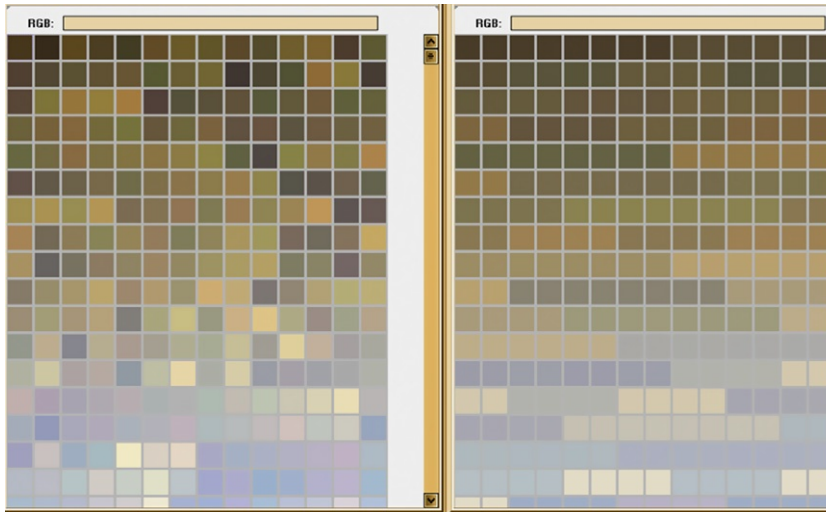

FIG. 20. Sorted palette before (left) and after (right) stego has been applied.

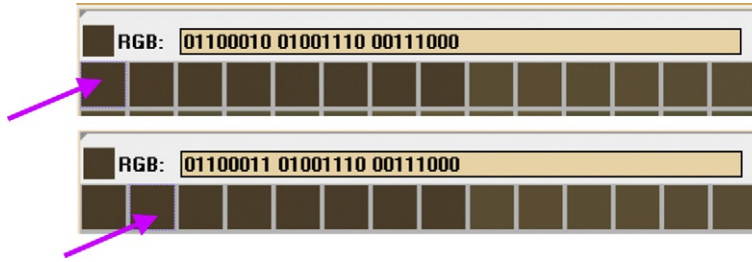

FIG. 21. Binary color values of adjacent palette entries in the stego image.

One caution in very small palette images that are used on the Web where only a handful of colors are used to display an icon-like object is that close color pairs can occur naturally. These images, however, would not be suitable for carrying hidden payloads of any size or value.

4.6 Format Modification

Format modification methods are a form of text semagram, where simple modifications are made to the carrier file containing the hidden text. Consider, for example, the following portion of text:

Enriched Air Nitrox (Wikipedia)
Nitrox refers to any gas mixture composed (excluding trace gases) of nitrogen and oxygen; this includes normal air which is approximately 78% nitrogen and 21% oxygen, with around 1% inert gases, primarily argon. However, in SCUBA diving, nitrox is normally differentiated and handled differently from air. The most common use of nitrox mixtures containing higher than normal levels of oxygen is in SCUBA, where the reduced percentage of nitrogen is advantageous in reducing nitrogen take up in the body's tissues and so extending the possible dive time, and/or reducing the risk of decompression sickness (also known as the bends).
Nitrox is mainly used in scuba diving to reduce the proportion of nitrogen in the breathing gas mixture. Reducing the proportion of nitrogen by increasing the proportion of oxygen reduces the risk of decompression sickness, allowing extended dive times without increasing the need for decompression stops. Nitrox is not a safer gas than compressed air in all respects: although its use reduces the risk of decompression sickness, it increases the risk of oxygen toxicity and fire, which are further discussed below.

A program called Snow can be used to hide a short text string into a text carrier file. In this example, the following text string has been placed into the carrier text file above:

We need to meet ASAP

Using a program called Snow to hide this string into the text block above would result in the file shown in Fig. 22, as seen in Word using *show* mode. Note all the extraneous space and tab characters; it is in these white space characters that the message is hidden.

4.7 Covert Communication Channels

Steganography itself forms a covert communication channel since, presumably, only the sender and receiver know where the hidden messages have been placed and how they can be retrieved. Some forms of stego actually employ covert communication channels within real-time data and network communications by exploiting the protocols themselves.

4.7.1 Stego and Communication Protocols

As the plethora of communication protocols evolves, the ability to embed (hide) information in these data streams in order to either leak information or to covertly communicate using these streams is possible. As with other forms of steganography, the first key to identifying the covert channels is to be looking for them.

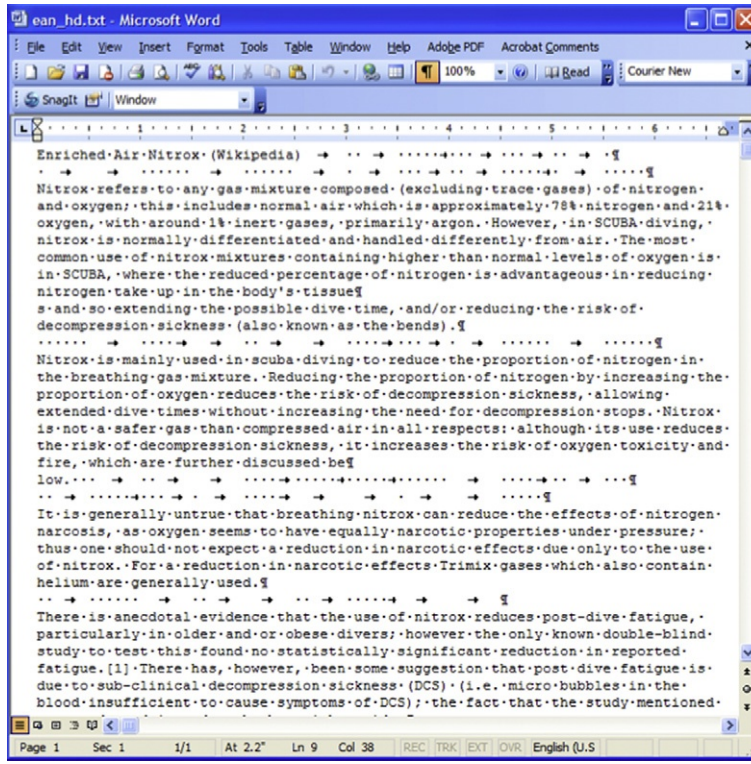

FIG. 22. Resultant text file after hiding a text string into a text file using the program Snow.

It is important to remember that individuals would use this method to leak information or covertly communicate when proven cryptography methods do not exist. Consider that the primary purpose for crypto is to deliver private and confidential communication between users that possess the proper credentials and keying material. The purpose or intent of steganography, however, is to hide the very existence of the communication channel.

Given this distinction, covert channels attempt to circumvent organization security policies by exploiting legitimate communication channels [29]. Organizations today have large, complex network and communications infrastructures. Each provides a point of attack for insiders or infected systems to communicate covertly. Utilizing compromised images and multimedia files in conjunction with Internet, e-mail, and other common infrastructure services to push files that contain hidden content represents the simplest form of this attack. More complex forms involve the

modification of the communication channel itself in order to exploit unused spaces and attributes of the channels.

Even wireless local area networks (WLANs) are susceptible to such attacks [30]. One such example is the Frame Control (FC) field of IEEE 802.11 WLAN frame header. Manipulating rarely used bits in the FC field, such as More Frag, Retry, PwrMgt, or More Data, can provide single or multiple bit alterations in every frame and, thus, a low-bandwidth side communication channel.

The method of modifying communication packets to embed hidden information is not new. Covert TCP by Craig Rowland [31], for example, forms covert communication channels using the Identification field in IP packets or the Sequence Number field in TCP segments [4,32]. As new protocols are developed, rarely used fields or fields that contain limited value offer new applications for steganography. Whether these protocols are TCP, IP, or User Datagram Protocol (UDP) based, or whether the application is client/server or peer-to-peer, exploitation opportunities exist.

4.7.2 Streaming Channels

With the advent of multimedia streaming data for audio, video, or Voice-over-IP (VoIP), several researchers theorized that embedding steganography in such streams would be difficult, if not impossible. The reason most often given is that these protocols assume data loss as a normal part of the protocol; thus, loss of data is not only tolerated, it is expected. Missing a couple of packets or discarding corrupted packets has limited, and only momentary, effects on the experience of the listener or viewer of digital communication streams. Indeed, this is something that we are all familiar with, as with a momentary burst of static on a voice channel or a frozen or pixilated picture on a digital movie, viewing or communicating over streams.

One could jump to the conclusion that an effective jamming attack against both static steganography (images and audio files) as well as within streaming media would be to routinely inject steganography noise into the data stream using similar techniques defined above. If done properly, there would be little impact on the resulting data stream. This argument holds true for nonlossy compressed images and audio types. Injecting noise into lossless data types such as JPEG and MP3 carriers, however, noticeably degrades the carrier file quite quickly because each time an image is re-encoded, the quality is affected.

Some experiments conducted at WetStone Technology with a range of sample images show that degradation begins to become visible to the naked eye after 200–500 cycles, depending on image and noise insertion characteristics. Larger images (3 MB and larger) of outside scenery with high color counts can sustain up to a 1000 injections before noticeable distortion is apparent. However, if you have embedded

a compressed, encrypted steganography payload within the data stream and data loss or corruption occurs, it is likely that all information after either of these events will be lost.

4.7.3 VoIP-Based Steganography

An increasing threat today—hiding data in an IP-based voice or video call—is an outcome of the natural evolution of steganography [33]. The Real-time Transport Protocol (RTP), described in RFC 1889, is a transport protocol for real-time applications [34]. A successful RTP environment provides an end-to-end transport with the ability to transmit real-time data such as audio and video. RTP uses UDP as its end-to-end transport protocol. Within VoIP environments, RTP provides the channel for call traffic. Therefore, there are three likely candidates for embedding steganography within the VoIP model:

1. UDP datagram headers: Exploitation of unused or rarely used header fields.
2. RTP packet headers: Exploitation of unused or rarely used header fields.
3. Voice payload: When making VoIP calls, the analog voice signals are transformed into digital content using a coder/decoder (codec), which is responsible for the analog-to-digital (and reverse) conversion. The digitized voice is then compressed and, in most environments, encrypted. The voice compression method, much like JPEG compression described previously, is lossy; therefore, modifications due to steganography must be made after the lossy compression stage and prior to any encryption.

All these general approaches suffer from potential data loss, a condition that must be overcome prior to implementing steganography within any of these channels. The first option is to employ error correction methodologies that will automatically correct a small number of lost or corrupted packets. These methods work quite well for normal channel traffic provided the payload is broken into multiple pieces. For example, it would not be wise to attempt transmission of a large compressed encrypted document over anything less than a perfect VoIP connection unless the payload was first broken into small discrete components. Error correction methodologies can be successfully applied to VoIP-based steganography, especially if the embedding takes place in the unused or rarely used bits in the packet header (this is because only a handful of bits are transmitted with each packet header). However, if voice payload steganography methods are employed, error correction techniques are less useful due to the amount of information embedded within each packet and the dependence on multiple consecutive packets being delivered without error.

A second method to overcome the data loss and corruption issue is to use encoded voice content to transfer data that is also resilient to data loss. For example, if audio

or video information is embedded into the VoIP channel, then losing a small percentage of packets will have limited impacts on the hidden payload. Some of the voice or video content might be lost, much like normal calls, but the message can still be communicated.

With the almost ubiquitous proliferation of VoIP, the ability to covertly communicate over these channels is quickly becoming a reality. With Android phones being delivered ready for custom application development, along with a growing Open Source VoIP community, the ability for both the good guys and the bad guys to exploit these devices for their own purposes is endless. The detection, cracking, and jamming of steganography laced covert communication channels are not at the end of a life-cycle, but rather just at the beginning. With the almost limitless number of VoIP calls, streaming audio and video content, and connected mobile devices, our ability to overtly or covertly communicate to anyone, anywhere, anytime is upon us. The question is what we will choose to do with it.

4.8 Conclusion and Summary

Gif-It-Up, JPHS, Snow, and S-Tools are used above, for example, purposes only; they are free, easy to use, and perform their tasks well. There are many other programs that can be used to hide information in BMP, GIF, JPEG, MPEG-1 Audio Layer 3 (MP3), Paintbrush (PCX), Portable Network Graphics (PNG), Tag Image File Format (TIFF), WAV, and other carrier file types. The StegoArchive.Com Web site has a good list of more than 400 freeware, shareware, and commercial steganography software for DOS, Linux/Unix, MacOS, Windows, and other operating systems [35].

There are many other ways in which messages can be hidden in digital media. Digital forensic examiners are very familiar with data that remains in file slack or unallocated space as the remnants of previous files and, of course, programs can be written that can access slack and unallocated space directly. Small amounts of data can also be hidden in the unused portion of file headers [14].

Information can also be hidden on a hard drive in a secret partition. A hidden partition will not be seen under normal circumstances although disk configuration and other tools might allow complete access to the hidden partition [4]. This theory has been implemented in a steganographic ext2fs file system for Linux. A hidden file system is particularly interesting because it protects the user from being inextricably tied to certain information on their hard drive. This form of *plausible deniability* allows a user to claim to not be in possession of certain information or to claim that certain events never occurred. Under this system, users can hide the number of files on the drive, guarantee the secrecy of the files' contents, and not disrupt nonhidden files by the removal of the stego file driver [12,36,37].

There are also several characteristics of sound that can be altered in ways that are indiscernible to human senses, and these slight alterations—such as tiny shifts in phase angle, speech cadence, and frequency—can transport hidden information [14].

Newer, more complex, steganography methods continue to emerge. *Spread spectrum steganography* methods are analogous to spread spectrum radio transmissions (developed in World War II and commonly used in data communication systems today) where the energy of the signal is spread across a wide frequency spectrum rather than focused on a single frequency, in an effort to make detection and jamming of the signal harder. Spread spectrum stego has the same function; avoid detection. These methods take advantage of the fact that little distortions to image and sound files are least detectable in the high-energy portions of the carrier; that is, high intensity in sound files or bright colors in image files. Even when viewed side by side, it is easier to fool human senses when small changes are made to loud sounds and/or bright colors [6].

5. Detecting Steganography

This section will discuss issues related to the detection of steganography software that hides information in digital files and carrier files that contain hidden information. Stego will be described in terms of the Prisoner's Problem. Steganalysis methods and tools will then be discussed.

5.1 The Prisoner's Problem

The Prisoner's Problem [38] is sometimes used to explain steganography in practice although it was originally introduced to describe a cryptography scenario. The problem involves two prisoners, Alice and Bob, who are locked in separate prison cells and wish to communicate with each other. Alice and Bob are allowed to exchange messages but William, the warden, can read all of the messages. Alice and Bob know that William will terminate the communications if he discovers the secret channel [39,40].

William can act in either a passive or active mode. In the *passive warden* model, William examines each message and determines whether to forward the message or not based upon his ability to detect a hidden message. In the *active warden* model, William can modify messages if he wishes. A conservative, or malicious, warden might actually modify all messages in an attempt to disrupt any covert channel so that Alice and Bob would need to use a very robust stego method [39,40].

The difficulty of the warden's task will depend largely on the complexity of the steganography algorithm and the amount of William's *a priori* knowledge [17,39,40]:

- In a *pure steganography* model, William knows nothing about the steganography method employed by Alice and Bob. This is a poor assumption on Alice and Bob's part as "security through obscurity" rarely works and is particularly disastrous when applied to cryptography. This is, however, often the model of the digital forensics analyst searching a Web site or hard drive for the possible use of steganography.
- *Secret key steganography* assumes that William knows the stego algorithm but does not know the secret stego/crypto key employed by Alice and Bob. This is wholly consistent with the assumption that a user of cryptography should make, per Kerckhoff's Principle; that is, "the security of the crypto scheme is in key management, *not* secrecy of the algorithm" [5]. This may also be too strong of an assumption for practice, however, as complete information would include access to the carrier file source.

5.2 Steganalysis Overview

Steganalysis, the detection of steganography by a third party, is a relatively young research discipline with few articles appearing before the late-1990s. The art and science of steganalysis is intended to detect (or estimate) hidden information based upon observing some data transfer, making no assumptions about the stego algorithm [39]. Detection of hidden data may not be sufficient; the steganalyst may also want to extract the hidden message, disable the hidden message so that the recipient cannot extract it, and/or alter the hidden message to send misinformation to the recipient [41]. Stego detection and extraction is generally sufficient if the purpose is evidence gathering related to a past crime, although destruction and/or alteration of the hidden information might also be legitimate law enforcement goals during an ongoing investigation of criminal or terrorist groups.

Steganalysis techniques can be classified in a similar way as cryptanalysis methods, largely based upon how much *a priori* information is known [14,20]:

- *Stego-only attack:* The stego medium is the only item available for analysis.
- *Known carrier attack:* The carrier and stego media are both available for analysis.
- *Known message attack:* The hidden message is known.
- *Chosen stego attack:* The stego medium and algorithm are both known.

- *Chosen message attack:* A known message and stego algorithm are used to create stego media for future analysis and comparison.
- *Known stego attack:* The carrier and stego medium, as well as the stego algorithm, are known.

Stego methods for digital media can be broadly classified as operating in the *image domain* or *transform domain*. Image domain tools hide the message in the carrier by some sort of bit-by-bit manipulation, such as LSB insertion. Transform domain tools manipulate the stego algorithm and the actual transformations employed in hiding the information, such as the DCT coefficients in JPEG images [20].

It follows, then, that steganalysis broadly follows the way in which the stego algorithm works. One simple approach is to visually inspect the carrier and stego media. Many simple stego tools work in the image domain and choose message bits in the carrier independently of the content of the carrier; while it is easier to hide the message in the area of brighter color or louder sound, the program may not seek those areas out. Thus, visual inspection may be sufficient to cast suspicion on a stego medium [6].

A second approach is to look for structural oddities that suggest manipulation. LSB insertion in a palette-based image often causes a large number of "duplicate" colors, where identical (or nearly identical) colors appear twice in the palette and differ only in the LSB. Stego programs that hide information merely by manipulating the order of colors in the palette cause structural changes, as well. The structural changes often create a signature of the stego algorithm that was employed [6,41].

Steganographic techniques generally alter the statistics of the carrier and, obviously, longer hidden messages will alter the carrier more than shorter ones [27,42–44]. Statistical analysis is commonly employed to detect hidden messages, particularly when the analyst is working in the blind [41]. There is a large body of work in the area of statistical steganalysis.

Statistical analysis of image and audio files can show whether the statistical properties of the files deviate from the expected norm [42,44,45]. These so-called first-order statistics—means, variances, chi-square (χ^2) tests—can measure the amount of redundant information and/or distortion in the medium. While these measures can yield a prediction as to whether the contents have been modified or seem suspicious, they are not definitive [6].

Statistical steganalysis is made harder because some stego algorithms take pains to preserve the carrier file's first-order statistics to avoid just this type of detection. Encrypting the hidden message also makes detection harder because encrypted data generally has a high degree of randomness, and ones and zeroes appear with equal likelihood [42,45].

Recovery of the hidden message, of course, adds another layer of complexity compared to merely detecting the presence of a hidden message. Recovering the

message requires knowledge or an estimate of the message length and, possibly, an encryption key and knowledge of the crypto algorithm [40].

Carrier file type-specific algorithms can make the analysis more straightforward. JPEG, in particular, has received a lot of research attention because of the way in which different algorithms operate on this type of file. JPEG is a poor carrier medium when using simple LSB insertion because the modification to the file caused by JPEG compression eases the task of detecting the hidden information [27]. There are several algorithms that hide information in JPEG files and all work differently; JSteg sequentially embeds the hidden data in LSBs, JPHS uses a random process to select LSBs, F5 uses a matrix encoding based on a Hamming code, and OutGuess preserves first-order statistics [17,45–49].

More advanced statistical tests using higher-order statistics, linear analysis, Markov random fields, wavelet statistics, and more on image and audio files have been described [42–44,50]. Detailed discussion is beyond the scope of this chapter, but the results of this research can be seen in some steganography detection tools.

Most steganalysis today is signature-based, similar to anti-virus and intrusion detection systems; anomaly-based steganalysis systems are just beginning to emerge. While the former systems are accurate and robust, the latter will be more flexible and better able to quickly respond to new stego techniques. One form of so-called blind steganography detection distinguishes between clean and stego images using statistics based on wavelet decomposition, or the examination of space, orientation, and scale across subsets of the larger image [41,42].

This type of statistical steganalysis is not limited to image and audio files. The Hydan program retains the size of the original carrier but, by using sets of functionally equivalent instructions, employs some instructions that are not commonly used. This opens Hydan to detection when examining the statistical distribution of a program's instructions.

The law enforcement, intelligence, and military communities do not always have the luxury of knowing either when and where stego has been used or the algorithm that has been employed. Generic tools that can detect—and classify—steganography is where research is still in its infancy but methods are already becoming available in software tools, some of which are described in the Section 5.3 [51].

And the same cycle is recurring as seen in the crypto world—steganalysis helps find embedded stego but also shows writers of new stego algorithms how to avoid detection.

5.3 Steganalysis of JPEG Images

Steganalysis can have a wide range of meanings depending upon whom you might be talking with, and there is certainly no single method or process. To explore some of the ways that steganalysis is actually performed, the discussion needs to narrow in

a carrier type so as to move from the general to the specific. In this section, a presentation will be made about detailed analysis of JPEG files.

Figure 23 shows the most common method of applying stego to an existing JPEG file by making modifications to the quantized DCT values. These values are, in essence, the result of the lossy compression stage and provide the input to the final lossless compression stage. Making careful and slight alterations to these values can deliver visually indistinguishable changes to the resulting image when decoded, although this does not mean they are undetectable.

Figure 24 depicts two JPEG images; the one on the left is the original carrier file and the one on the right is the same image where data have been hidden using the JPEG HiIde&Seek (JPHS) stego software. When performing steganalysis, the examiner most likely will have only one image (assuming that the suspect is a professional and has been careful to keep the original image secluded); to illustrate the challenge, however, two images will be employed.

Examining the properties of the two images immediately shows several differences, namely, file size, used colors, and maximum payload. Figure 25 shows the JPEG headers and shows additional modifications that have occurred; in the stego image, in particular, the JPEG header information has been stripped out. There are two schools of thought on this from the steganographer's perspective. The first, and predominant, view is to strip the headers to eliminate any pedigree information that might be present in the header, such as identity, camera, photo source, etc. The second school of thought says to leave the header information intact so as to make the stego image look as conventional as possible.

The primary goal of any stego program is to ensure that the image, audio file, movie, or data stream looks, sounds, and behaves as close to normal as possible. Thus, visually examining a JPEG image may seem futile. However, by rendering the resulting JPEG with certain filters, anomalies caused by the stego process may

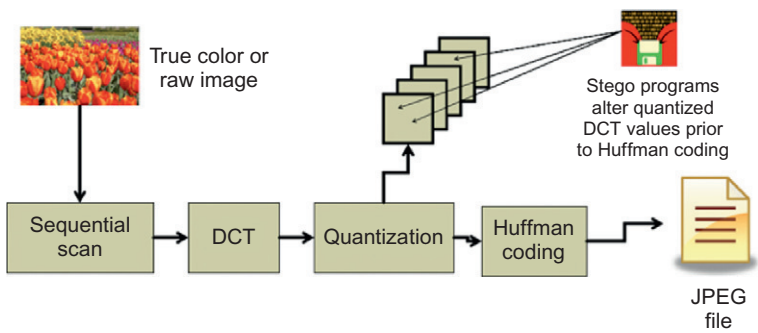

FIG. 23. JPEG encoding.

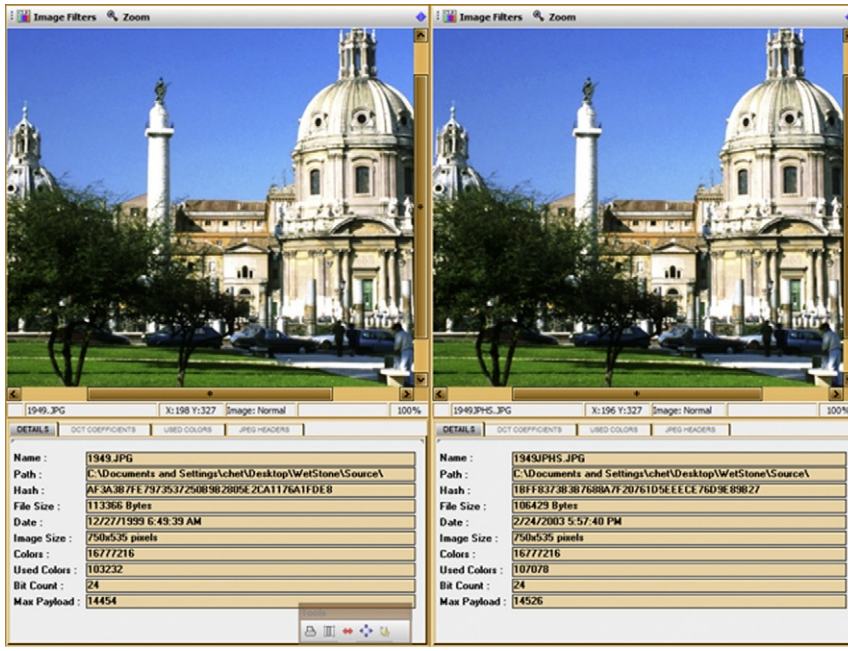

FIG. 24. Original JPEG image carrier (left) and image with hidden data (right).

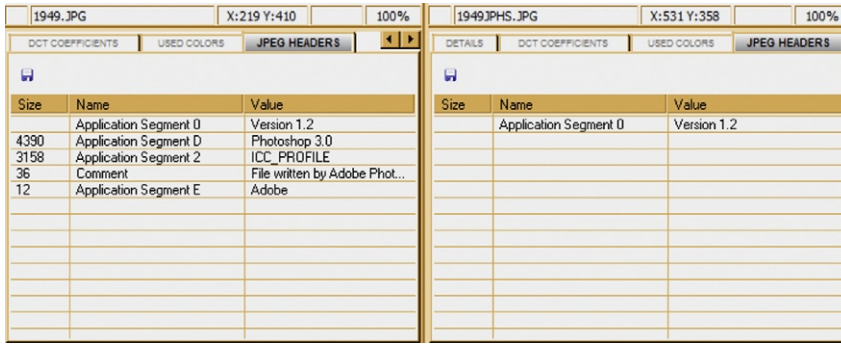

FIG. 25. JPEG header analysis (original carrier on left, stego image on right).

emerge. Figure 26, for example, shows the result when rendering the JPEG images using a hue filter. The reader can clearly see the distortion effect in the stego image on the right versus the original on the left, particularly in the area of the sky. Similar results would occur when filtering based on saturation or intensity.

These artifacts are caused by the modification of the quantized DCT values during the stego insertion. When the image is rendered (i.e., decoded from JPEG to RGB), the hue angles have been distorted from the normal state found in the original image. The amount of distortion is directly related to how much information is embedded; in other words, the size of the hidden payload in relationship to the content in the original image. This effect is a second-order artifact since these resulting RGB effects are caused by alterations that have been made to the quantized DCT values.

Direct examination of the quantized DCT values is also possible. Figure 27 shows a histogram of the discrete coefficients (DC) of both the original (left) and stego (right) image. At first glance, the two histograms look similar in shape. Closer examination, however, shows very discernable and measurable differences.

Figure 28 narrows the focus of the analysis to the peak areas of the two histograms, in this case, DC value is 169. Notice that the value 169 occurred as a DC coefficient 874 times in the original image (left), with the adjacent values of 168 and 170 occurring 25 and 0 times, respectively. Knowing that a key element of JPEG compression is the preparation of the quantized DC value to aid in the lossless Huffman compression that follows this type of statistical distribution makes sense, especially near the peak values. However, the examination of the stego image (right) shows that the 169 value has been decreased to only 423 occurrences, while the 168 value has increased to 426. This change is the equivalent of a least significant bit

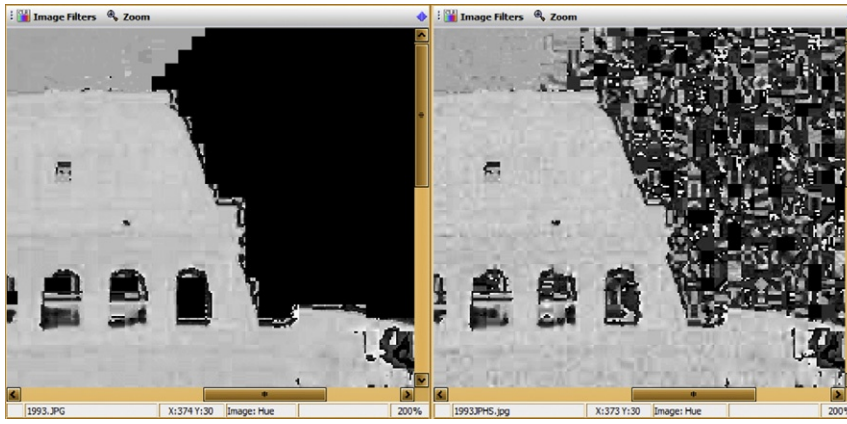

FIG. 26. JPEG hue rendering (original carrier on left, stego image on right).

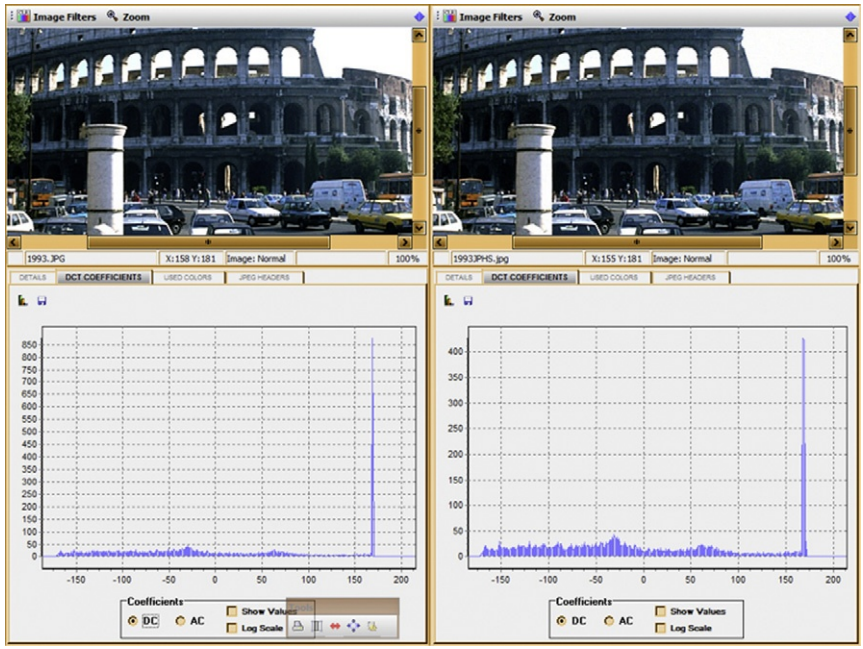

FIG. 27. Quantized DCT histogram (original carrier on left, stego image on right).

change used to hide bits of the hidden payload. This slight change has minimal impact on the normal rendering of the image but is clearly visible here as an anomaly that is a direct indicator that stego may be in play.

Even in a simple analysis of an image, the process can be both time consuming and may change or vary based on each result. It is important to point out that even when employing steganography detection algorithms to identify high probability images, several of the examination steps above must be applied to validate the results or rule out possible false positives. It is true that some signature detection capabilities do exist for stego where we can identify specific characteristics of well-known stego embedding programs, but for the more advanced methods where anomaly detection methods must be employed, human analysis and examination of the resulting images is an important step to confirm the presence of hidden evidence. Of course, the ultimate confirmation comes from the cracking of the known stego and the ultimate recovery of the hidden payload. Detailed analysis of those techniques is beyond the scope of this chapter.

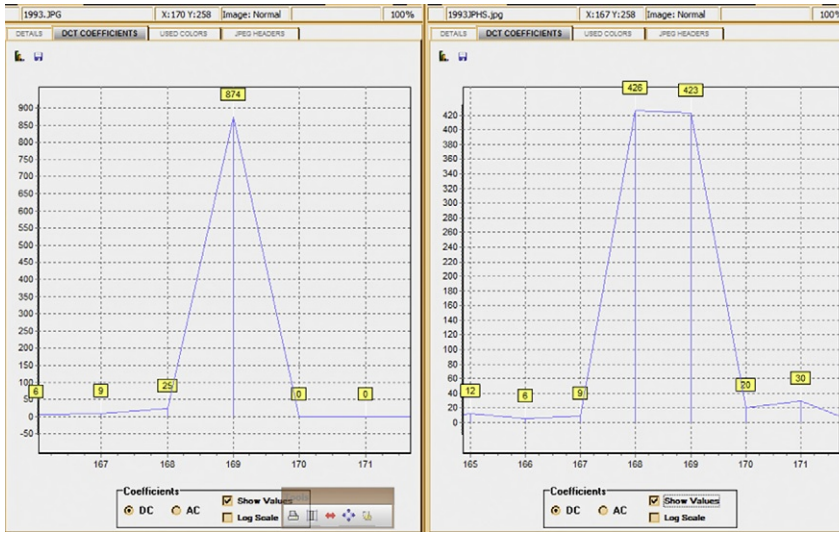

FIG. 28. Quantized DCT analysis (original carrier on left, stego image on right).

6. Steganography Detection Tools

This chapter has a stated focus on the practicing computer forensic examiner rather than the researcher. This section, then, will show some examples of currently available software that can detect the presence of stego programs, detect suspect carrier files, and disrupt steganographically hidden messages. This is by no means a survey of all available tools, but an example of available capabilities; StegoArchive.com lists many steganalysis programs [35], and Hayati et al.[52] provide an overview of over 100 different stego and steganalysis software packages.

6.1 Steganography Detection Tools

The detection of stego software on a suspect computer is important to the subsequent forensic analysis. Many stego detection programs work best when there are clues as to the type of stego that was employed in the first place. Finding stego software on a computer would give rise to the suspicion that there are actually stego files with hidden messages on the suspect computer. Further, the type of stego software found will directly impact any subsequent steganalysis; for example,

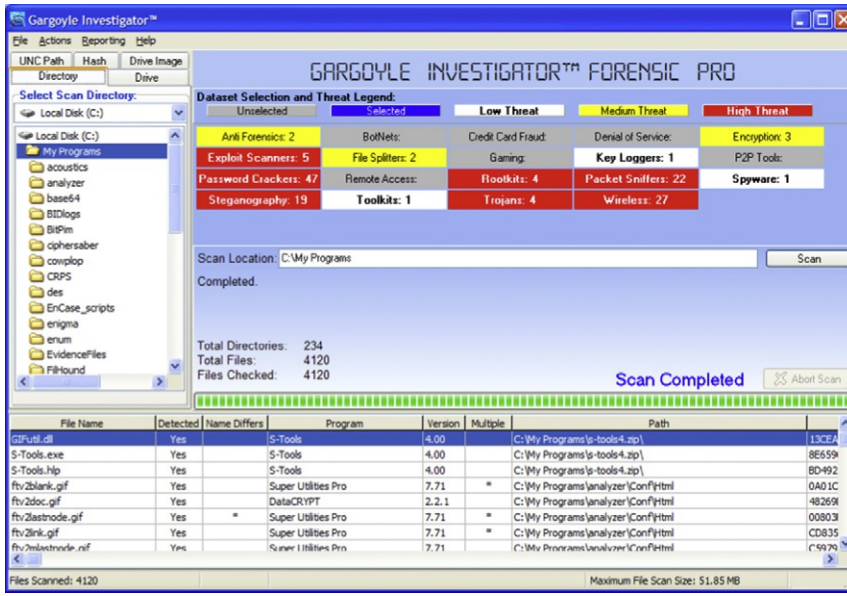

FIG. 29. The output from Gargoyle when aimed at one of the directories on author Kessler's hard drive.

S-Tools might direct one's attention to GIF, BMP, and WAV files while JPHS might direct the analyst to look more closely at JPEG files.

WetStone Technologies' Gargoyle software [53] can be used to detect the presence of stego software and other forms of malware. Gargoyle employs a proprietary hash set of all the files in the known stego software distributions, comparing them to the hashes of the files subject to search. Figure 29 shows the output when Gargoyle was directed to target one of the directories on the author's systems where stego programs are stored. Gargoyle data sets can also be used to detect the presence of cryptography, instant messaging, key logging, Trojan horse, password cracking, and other malicious software. WetStone's Stego Hunter, part of WetStone's Stego Suite [54], is designed to detect remnants of steganography programs. Stego Hunter can be directed to examine a local hard drive, or mount and examine an E01 (EnCase), AD1 (AccessData), dd, ISO, or other image file (Fig. 30).

Similar functionality can be found in the Steganography Analyzer Artifact Scanner (StegAlyzerAS) from Steganography Analysis and Research Center (SARC) [55] (Fig. 31). StegAlyzerAS uses SARC's Steganography Application Fingerprint Database (SAFDB) to detect the presence of stego application artifacts.

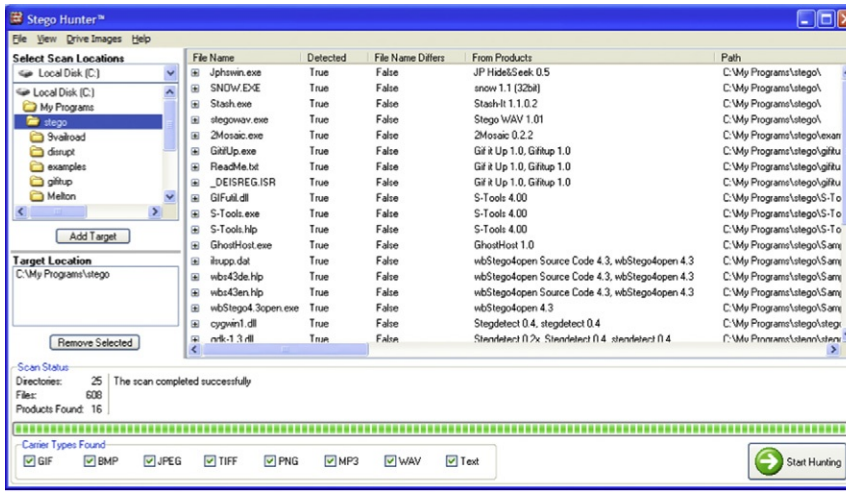

FIG. 30. The output from Stego Hunter when aimed at one of the directories on author Kessler's hard drive.

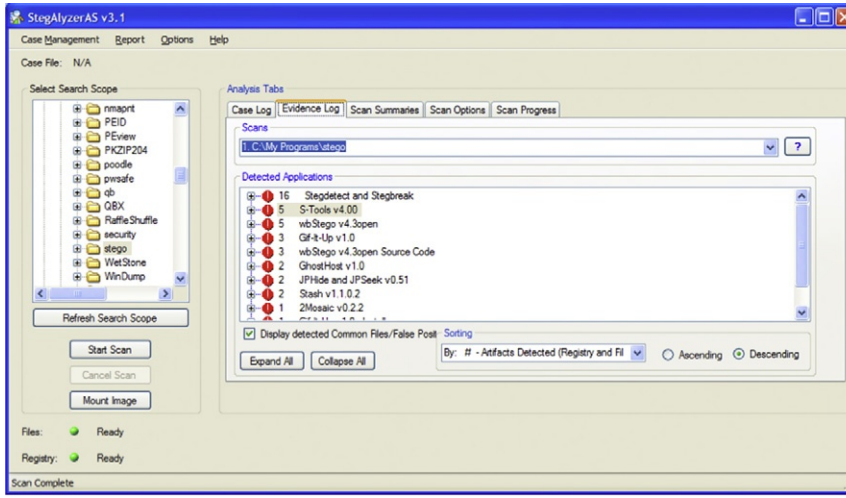

FIG. 31. The output from StegAlyzerAS when aimed at one of the directories on author Kessler's hard drive.

The detection of steganography software is a hard problem due to the small size of the software coupled with the increasingly large storage capacity of removable media. S-Tools, for example, requires less than 600 KB of disk space and can be executed directly, without additional installation, from a USB memory key. Under those circumstances, no remnants of the program would be found on the hard drive.

6.2 Stego Carrier File Detection

The second important function of steganography detection software is to find possible carrier files. Ideally, the detection software would also provide some clues as to the stego algorithm used to hide information in the suspect file so that the analyst might be able to attempt recovery of the hidden information.

One commonly used detection program is Niels Provos' stegdetect. Stegdetect can find hidden information in JPEG images using such stego schemes as F5, Invisible Secrets, JPHide, and JSteg [56]. Figure 32 shows the output from xsteg, a graphical interface for stegdetect, when used to examine two files on one of the author's hard drive—the original carrier file and one containing the Burlington

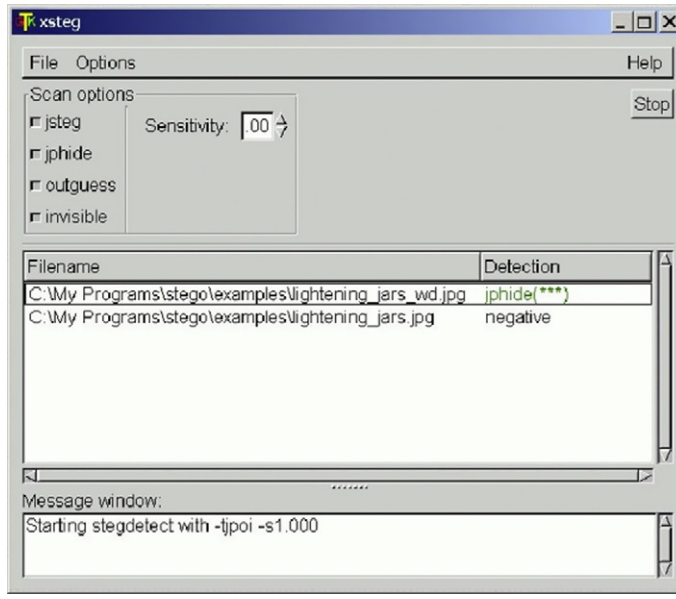

FIG. 32. The output from xsteg when examining two suspect JPEG files.

airport map. Note that the stego file is not only flagged as containing hidden information, but the program also suggests (correctly) the use of the JPHide stego scheme.

WetStone Technologies' Stego Suite [54] comprises four program that do the following:

- *Stego Hunter:* Detects remnants of steganography programs (described above)
- *Stego Watch:* Anomaly-based, blind stego detection software
- *Stego Analyst:* Imaging and analysis tool, providing visual clues that steganography may have been used in both image and audio files
- *Stego Break:* A password cracker for stego carrier files

Stego Watch analyzes a set of files and provides a probability about which are stego media and the likely algorithm that was used for hiding information; this metric, in turn, provides clues as to the most likely software employed. The analysis uses a variety of user-selectable statistical tests that are based upon the carrier file characteristics that might be altered by the different stego methods. Knowing the stego software that is available on the suspect computer will, obviously, help the analyst in selecting the most likely statistical tests. Figure 33 shows the output from

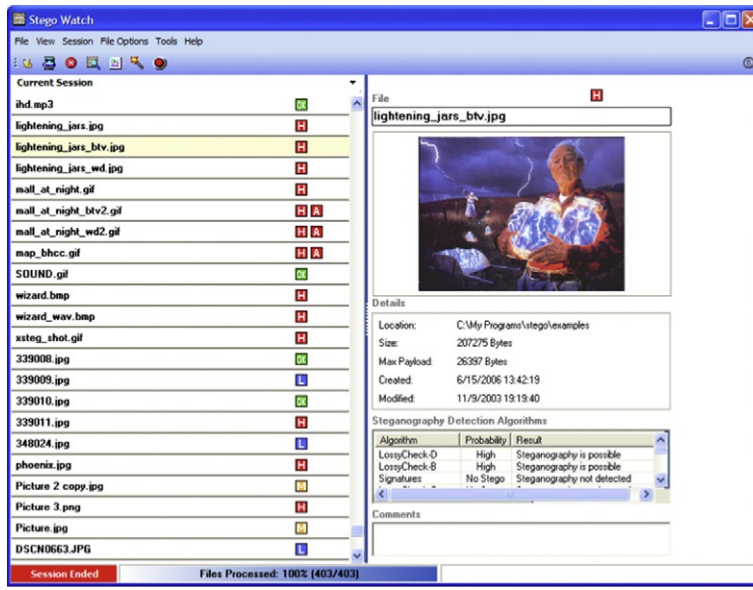

FIG. 33. Information from Stego Watch about a JPEG file suspected to be a stego carrier.

Stego Watch when aimed at the JPEG carrier file containing the airport map. Note that the Stego Detection Algorithms section in the display shows the statistical algorithms that were employed for analysis and the ones that bore fruit for this image; in this case, Stego Watch's LossyCheck-B and LossyCheck-D tests are highly suggestive of the use of the JPHide stego method which, in turn, suggests the use of the JPHide&Seek (JPHS) software.

Stego Analyzer provides a number of detailed analysis tools that allow the examination of image files for clues that something has been altered in the file. Alteration of an image can be suggestive of many things, including the possibility that data has been hidden. Stego Analyst can display many aspects of an image file, such as the palette, intensity, saturation, and hue of paletted images, or the DCT coefficients (Fig. 34), color set, and headers of JPEG files.

The SARC Steganography Analyzer Signature Scanner (StegAlyzerSS) [57] examines possible carrier files using three tests: a stego application signature scan (Fig. 35), a search for image file trailers to detect stego by appending hidden data, and an analysis searching for possible LSB overwriting methods.

Finding steganography in a file suspected to contain it is relatively easy compared to the extraction of the hidden data. Most stego software uses passwords for secrecy, randomization, and/or encryption. Stegbreak, a companion program to stegdetect,

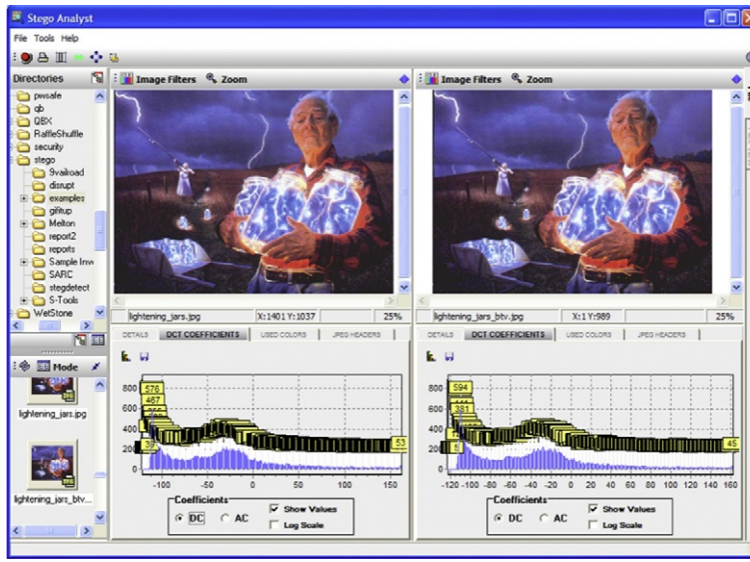

FIG. 34. DCT coefficient analysis of a JPEG carrier file with Stego Analyst.

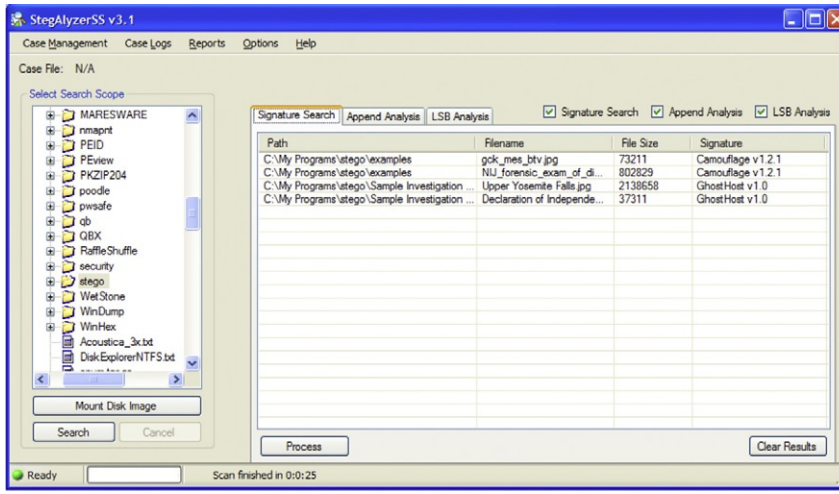

FIG. 35. Signature search display from StegAlyzerSS.

uses a dictionary attack against JSteg-Shell, JPHide, and OutGuess algorithms to find the password of the hidden data but, again, this is only applicable to JPEG files [56]. Similarly, Stego Break, a component part of WetStone's Stego Watch, uses a dictionary attack on suspect files in an attempt to extract and recover hidden data [54]. Steganography detection schemes do not directly help in the recovery of the password; finding appropriate clues is where the rest of the investigation and computer forensics comes into play.

The Prisoner's Problem suggests that a steganalyst may or may not want to disrupt the hidden data in a known carrier file. A computer forensic examiner looking at evidence in a criminal case probably has no reason to alter any evidence files. However, an examination that is part of an ongoing terrorist surveillance might well want to disrupt the hidden information even if it cannot be recovered. Hidden content, such as steganography and digital watermarks, can be attacked in several ways so that it can be removed or altered [58,59], and there is software specifically designed to attack digital watermarks. Such attacks have one of two possible effects: they either reduce the stego carrying capacity of the carrier (necessary to avoid the attack) or fully disable the capability of the carrier as a stego medium.

Although this subject is also beyond the scope of this chapter, one interesting example of stego disruption software can be used to close this discussion. 2 Mosaic, by Fabien Petitcolas, employs a so-called "presentation attack" primarily against images on a Web site. 2 Mosaic attacks a digital watermarking system by chopping

FIG. 36. A portion of the JPEG image with the hidden airport map, created by 2Mosaic.

an image into some number of smaller subimages. On the Web site, the series of small images are positioned next to each other and appear the same as the original large image [60].

Figure 36 shows an example of 2 Mosaic when used against the JPEG image from Fig. 13. In this case, the carrier file is split into 165 subimages, as shown in the figure (11 rows of 15 subimages). The 2 Mosaic approach of course is obvious when used; the viewer of the altered image knows immediately that something is amiss.

7. Summary and Conclusions

Consider the following hypothetical scenario. By preagreement among members of a terrorist organization, the leader puts an item for sale on an Internet auction site every Monday morning and posts a photo of the item. The item for sale is legitimate; bids are accepted, money is collected, and the item is dutifully delivered. At some prearranged time during the week, a version of the photo is posted that contains a hidden message. The cell members know when that time is and download the weekly message—and sometimes even bid on the item. As time goes on, the leader gets good ratings on the site and receives thousands of visitors each week; even if this site was under active surveillance, it is unclear that the few members of the cell can be distinguished from the thousands of legitimate bidders.

This scenario—or one like it—is a viable method for terrorists or criminals to communicate, but is it real? In the aftermath of 9/11, a number of articles appeared suggesting that al Qaeda terrorists employ stego [51,61–63] and alleged Russian spies arrested in the United States in the summer of 2010 reportedly hid communications in images on public Web sites [64,65]. In partial response to the earlier

reports, several attempts have been made to ascertain the presence of stego images on the Internet. One well-known study searched more than three million JPEG images at eBay and USENET archives; using stegdetect, 1–2% of the images were found to be "suspicious," but no hidden messages were recovered using stegbreak [17,45]. Another study examined several hundred thousand images from a random set of Web sites and, also using stegdetect and stegbreak, obtained similar results [66].

While these projects do provide a framework for searching a Web site for stego images, no conclusions can be drawn from them about stego images on the Internet. First and foremost, stegdetect only looks at JPEG images; other image types were never examined. Second, a very limited number of Web sites were examined, too few to make any definitive statements about the Internet as a whole. It is also interesting to note that several stego researchers are purposely not publishing information about what Internet sites they are examining or what they are finding [51,62].

There appear to be few hard statistics about the frequency with which steganography software or media are discovered by law enforcement officials in the course of computer forensic examinations. Anecdotal evidence suggests, however, that many computer forensic examiners do not routinely search for stego software and many might not recognize such tools if they found them. In addition, the tools that are employed to detect stego software are often inadequate, frequently relying solely on hash sets or the stego tools themselves [67]. A thorough search for evidence of steganography on a suspect hard drive that might contain thousands of images, audio files, and video clips could take days [2].

Indeed, many digital forensic examiners consider the search for stego tools and/or stego media to be a routine part of every examination. But what appears to be lacking is a set of guidelines providing a systematic approach to stego detection; even the U.S. Department of Justice search and seizure guidelines for digital evidence barely mention steganography [68,69]. Steganalysis will only be one part of an investigation, however, and an investigator might need clues from other aspects of the case to point them in the right direction. A computer forensic examiner might suspect the use of stego because of the nature of the crime, books in the suspect's library, the type of hardware or software discovered, large sets of seemingly duplicate images, statements made by the suspect or witnesses, or other factors. A Web site might be suspect by the nature of its content or the population that it serves. These same items might give the examiner clues to passwords, as well. And searching for steganography is not only necessary in criminal investigations and intelligence gathering operations; forensic accounting investigators are realizing the need to search for stego as this becomes a viable way to hide financial records [2,70].

It is impossible to know for sure at this point how widespread the use of steganography is by criminals and terrorists [2]. Today's truth, however, may not even matter; the use of stego is sure to increase and will be a growing hurdle for law enforcement and antiterrorism activities. Ignoring the significance of stego because of the lack of statistics is "security through denial" and not a good strategy.

Stego will certainly not be found if it is not being looked for. And as the number of potential carriers grows, the opportunity for these covert channels rises exponentially.

Appendix A. Additional Web Sites

Computer Forensics, Cybercrime and Steganography Resources Web site, "Steganography & Data Hiding—Articles, Links, and Whitepapers" page (http://www.forensics.nl/steganography).

GCK's steganography links (http://www.garykessler.net/library/securityurl.html#crypto).

Neil Johnson's Steganography & Digital Watermarking page (http://www.jjtc.com/Steganography/).

Appendix B. Companion Downloads to this Chapter

The hidden, carrier, and stego files mentioned in this chapter can be downloaded from *http://www.garykessler.net/download/stego.zip*. Use the password *tyui* to recover the hidden file from the stego files. The contents of the ZIP file include:

- Figure 11, Airport image: *btv_map.gif*
- Figure 12a, Original WAV carrier file: *hitchhiker_beginning.wav*
- Figure 12b, Stego WAV file: *hitchhiker_beginning_btv.wav*
- Figure 13, JPEG carrier file: *lightening_jars.jpg*
- Figure 13, Stego JPEG file: *lightening_jars_btv.jpg*
- Figure 14, Original and stego JPEG file: *gck_mes.jpg* and *gck_mes_btv.jpg*
- Figure 16, Original GIF carrier file: *mall_at_night.gif*
- Figure 16, Stego GIF file: *mall_at_night_btv2.gif*
- Figure 36, Disrupted stego JPEG file: *disrupt/lighte~1.html*

The noncommercial software employed in the examples in this chapter can be downloaded from the following mirror site:

- 2 Mosaic (*http://www.garykessler.net/download/2Mosaic_0_2_2.zip*)
- Camouflage (*http://www.garykessler.net/download/Camou121.exe*)
- Gif-It-Up (*http://www.garykessler.net/download/Gif-it-up.exe*)
- JPHS for Windows (*http://www.garykessler.net/download/jphs_05.zip*)
- Snow (*http://www.garykessler.net/download/snow.zip*)
- Stegdetect (*http://www.garykessler.net/download/stegdetect-0.4.zip*)
- S-Tools (*http://www.garykessler.net/download/s-tools4.zip*)

References

[1] F.L. Bauer, Decrypted Secrets: Methods and Maxims of Cryptology, fourth ed., Springer-Verlag, Berlin, 2007.

[2] C. Hosmer, C. Hyde, Discovering covert digital evidence, Digital Forensic Research Workshop (DFRWS) 2003, (2003, August). Retrieved April 16, 2010, from http://www.dfrws.org/2003/presentations/Paper-Hosmer-digitalevidence.pdf

[3] M. Arnold, M. Schmucker, S.D. Wolthusen, Techniques and Applications of Digital Watermarking and Content Protection, Artech House, Norwood, MA, 2003.

[4] N.F. Johnson, Z. Duric, S.G. Jajodia, Information Hiding: Steganography and Watermarking—Attacks and Countermeasures, Kluwer Academic Publishers, Norwell, MA, 2001.

[5] D. Kahn, The Codebreakers: The Story of Secret Writing, revised ed., Scribner, New York, 1996.

[6] P. Wayner, Disappearing Cryptography—Information Hiding: Steganography & Watermarking, third ed., Morgan Kaufmann, Burlington, MA, 2009.

[7] Warchalking, Wikipedia, the Free Encyclopedia. Retrieved April 16, 2010, from http://en.wikipedia.org/wiki/Warchalking. 2010, January 31.

[8] M. Barni, C.I. Podilchuk, F. Bartolini, E.J. Delp, Watermark embedding: hiding a signal within a cover image, IEEE Commun. 39 (8) (2001, August) 102–108.

[9] S.H. Kwok, Watermark-based Copyright Protection System Security, Commun. ACM 46 (10) (2003, October) 98–101.

[10] M. Duren, M. Davis, C. Hosmer, Steganography vs. Digital Watermarking, in: The Science of Digital Investigation blog, (2008, September 24). Retrieved April 16, 2010, from http://www.wetstonetech.com/blogs/blog1.php/2008/09/24/test-post

[11] H. Clinton, Letter to John Burgoyne. Spy Letters of the American Revolution, from the Collection of the Clements Library, Retrieved April 17, 2010, from http://www.clements.umich.edu/Spies/letter-1777august10-1.html, 1777, August 10.

[12] D. Artz, Digital steganography: hiding data within data, IEEE Internet Comput. 5 (3) (2001, May/June) 75–80. Retrieved April 17, 2010, from http://www.cc.gatech.edu/classes/AY2003/cs6262_fall/digital_steganography.pdf

[13] The MathWorks, RGB Color Cube for `uint8` Images. Retrieved April 18, 2010, from http://www.mathworks.com/access/helpdesk/help/toolbox/images/color7.gif, 2010.

[14] K. Curran, K. Bailey, An evaluation of image based steganography methods, Int. J. Digital Evidence 2 (2) 2003, Fall. Retrieved April 17, 2010, from http://www.utica.edu/academic/institutes/ecii/publications/articles/A0AD276C-EACF-6F38-E32EFA1ADF1E36CC.pdf
[15] N.F. Johnson, S. Jajodia, Exploring steganography: seeing the unseen, IEEE Computer 31 (2) (1998, February) 26–34. Retrieved April 17, 2010, from http://www.jjtc.com/pub/r2026.pdf
[16] Monash University (n.d.). JPEG Image Coding Standard. Retrieved April 17, 2010, from http://www.ctie.monash.edu.au/emerge/multimedia/jpeg/.
[17] N. Provos, P. Honeyman, Hide and seek: an introduction to steganography, IEEE Security Privacy 1 (3) 2003, May/June. Retrieved April 17, 2010, from http://niels.xtdnet.nl/papers/practical.pdf
[18] R.F. Rey (Ed.), Engineering and Operations in the Bell System, AT&T Bell Laboratories, Murray Hill, NJ, 1983.
[19] B. Fries, M. Fries, The MP3 and Internet Audio Handbook, Tacoma Books, Burtonsville, MD, 2000.
[20] N.F. Johnson, S. Jajodia, Steganalysis of images created using current steganography software, D. Aucsmith (Ed.), in: Proceeding of the Second International Workshop on Information Hiding (IH '98), Lecture Notes in Computer Science, vol. 1525, Springer-Verlag, New York, 1998, April, pp. 273–289.Portland, OR. Retrieved April 19, 2010, from http://www.jjtc.com/ihws98/jjgmu.html
[21] G.C. Kessler, An overview of cryptography, GaryKessler.net Web site. Retrieved April 19, 2010, from http://www.garykessler.net/library/crypto.html, 2010, March 16.
[22] spam mimic, Spam Mimic. 2009.Web site. Retrieved April 17, 2010, from http://www.spammimic.com/
[23] R. El-Khalil hydan. Retrieved April 19, 2010, from http://www.crazyboy.com/hydan/
[24] R. El-Khalil, A.D. Keromytis, Hydan: Hiding Information in Program Binaries, Proceedings of the 6th International Conference on Information and Communications Security, 2004, pp. 187–199. Retrieved April 19, 2010, from http://www1.cs.columbia.edu/~angelos/Papers/hydan.pdf
[25] G.C. Kessler, File signatures table, GaryKessler.net Web site. Retrieved April 19, 2010, from http://www.garykessler.net/library/file_sigs.html, 2010, April 15.
[26] S. Inch, A simple image hiding technique: what you may be missing, J. Digital Forensic Pract. 2 (2) (2008, April) 83–94.
[27] J. Fridrich, R. Du, Secure steganographic methods for palette images, Proceedings of the 3rd Information Hiding Workshop, Lecture Notes in Computer Science, vol. 1768, Springer-Verlag, New York, 1999, September, pp. 47–60. Dresden, Germany. 2000. Retrieved April 15, 2010, from http://www.ws.binghamton.edu/fridrich/Research/ihw99_paper1.dot.
[28] T.A. Welch, A technique for high-performance data compression, IEEE Computer 17 (6) (1984) 8–19. Retrieved June 7, 2010, from http://www.cs.duke.edu/courses/spring03/cps296.5/papers/welch_1984_technique_for.pdf
[29] R.C. Newman, Covert Computer and Network Communications, in: Proceedings of the 4th Annual Conference on Information Security Curriculum Development 2007, September 28–29, 2007.
[30] C. Krätzer, J. Dittmann, A. Lang, T. Kühne, WLAN steganography: a first practical review, Proceedings of the 8th Workshop on Multimedia and Security (MM&Sec '06), 2006, pp. 17–22. September 26–27, 2006, Geneva, Switzerland.
[31] C.H. Rowland, Covert channels in the TCP/IP protocol suite, First Monday 2 (5) 1996. Retrieved April 15, 2010, from http://131.193.153.231/www/issues/issue2_5/rowland/index.html
[32] S.J. Murdoch, S. Lewis, Embedding covert channels into TCP/IP, in: Proceedings of the 7th Information Hiding Workshop, 2005, June. Barcelona, Italy. Retrieved April 15, 2010, from http://www.cl.cam.ac.uk/~sjm217/papers/ih05coverttcp.pdf

[33] W. Mazurczyk, K. Szczypiorski, Steganography of VoIP Streams, in: R. Meersman, Z. Tari (Eds.), Proceedings of On the Move to Meaningful Internet Systems: OTM 2008, 2008, pp. 1001–1018. Part II, LNCS 5332.

[34] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, Internet Engineering Task Force, Request for Comments (RFC) 1889, January.

[35] StegoArchive.com, Stego Archive Web site. Retrieved April 15, 2010, from http://home.comcast.net/~ebm.md/stego.html, 2005.

[36] R. Anderson, R. Needham, A. Shamir, The Steganographic File System, in: D. Aucsmith (Ed.), Proceedings of the Second International Workshop on Information Hiding (IH '98), Portland, OR, Lecture Notes in Computer Science, vol. 1525, Springer-Verlag, New York, 1998, April, pp. 73–82. Retrieved April 15, 2010, from http://www.cl.cam.ac.uk/~rja14/Papers/stego-fs.pdf

[37] A.D. McDonald, M.G. Kuhn, StegFS: a steganographic file system for Linux, A. Pfitzmann (Ed.), Proceedings of the Third International Workshop on Information Hiding (IH '99), Lecture Notes in Computer Science, vol. 1768, Springer-Verlag, New York, 1999, September-October, pp. 462–477. Dresden, Germany. Retrieved April 15, 2010, from http://www.cl.cam.ac.uk/~mgk25/ih99-stegfs.pdf

[38] G.J. Simmons, The prisoners' problem and the subliminal channel, in: D. Chaum (Ed.), Advances in Cryptology: Proceedings of CRYPTO 83, Santa Barbara, CA, Plenum Press, New York, 1984, pp. 51–67.

[39] R.A. Chandramouli, Mathematical approach to steganalysis, in: E.J. Delp III, P.W. Wong (Eds.), Proceedings of SPIE Security and Watermarking of Multimedia Contents IV, 2002, April, pp. 14–25. San Jose, CA, January 2002. Retrieved April 15, 2010, from http://www.ece.stevens-tech.edu/~mouli/spiesteg02.pdf

[40] J. Fridrich, M. Goljan, D. Hogea, D. Soukal, Quantitative steganalysis of digital images: estimating the secret message length, ACM Multimedia Syst. J. 9 (3) (2003, September) 288–302. Special issue on Multimedia Security, Retrieved April 15, 2010, from http://www.ws.binghamton.edu/fridrich/Research/mms100.pdf

[41] J.T. Jackson, G.H. Gunsch, R.L. Claypoole, G.B. Lamont, Blind steganography detection using a computational immune system: a work in progress, Int. J. Digital Evidence 1 (4) 2003, Winter. Retrieved April 19, 2010, from http://www.utica.edu/academic/institutes/ecii/publications/articles/A04D31C4-A8D2-ADFD-E80423612B6AF885.pdf

[42] H. Farid, Detecting Steganographic Messages in Digital Images, Technical Report TR2001-412 Dartmouth College, Computer Science Department, Hanover, NH, 2001. Retrieved April 19, 2010, from http://www.cs.dartmouth.edu/~farid/publications/tr01.pdf

[43] J. Fridrich, M. Goljan, Practical steganalysis—state of the art, in: E.J. Delp III, P.W. Wong (Eds.), Proceedings of SPIE Security and Watermarking of Multimedia Contents IV, 2002, April, pp. 1–13. San Jose, CA, January 2002. Retrieved April 15, 2010, from http://www.ws.binghamton.edu/fridrich/Research/steganalysis01.pdf

[44] H. Özer, İ. Avcıbaş, B. Sankur, N. Memon, Steganalysis of Audio Based on Audio Quality Metrics, in: E.J. Delp III, P.W. Wong (Eds.), Proceedings of SPIE Security and Watermarking of Multimedia Contents V, 2003, June, pp. 55–66. Santa Clara, CA, January 2003. Retrieved April 18, 2010, from http://www.busim.ee.boun.edu.tr/~sankur/SankurFolder/Audio_Steganalysis_16.doc

[45] N. Provos, P. Honeyman, Detecting Steganographic Content on the Internet, Center for Information Technology Integration, University of Michigan, 2001, August 31.CITI Technical Report 01–11. Retrieved April 17, 2010, from http://www.citi.umich.edu/techreports/reports/citi-tr-01-11.pdf

[46] J. Fridrich, M. Goljan, R. Du, Steganalysis based on JPEG compatibility, in: A.G. Tescher, B. Vasudev, V.M. Bove Jr. (Eds.), Proceedings of SPIE Multimedia Systems and Applications IV,

2001, November, pp. 275–280. Denver, CO, August 2001. Retrieved April 19, 2010, from http://www.ws.binghamton.edu/fridrich/Research/jpgstego01.pdf

[47] J. Fridrich, M. Goljan, D. Hogea, Steganalysis of JPEG images: breaking the F5 algorithm, 5th Information Hiding Workshop, 2002, October. Noordwijkerhout, The Netherlands. Retrieved April 19, 2010, from http://www.ws.binghamton.edu/fridrich/Research/f5.pdf

[48] J. Fridrich, M. Goljan, D. Hogea, Attacking the OutGuess, Proceedings of the ACM Workshop on Multimedia and Security 2002, 2002, December. Juan-les-Pins, France. Retrieved April 19, 2010, from http://www.ws.binghamton.edu/fridrich/Research/acm_outguess.pdf

[49] J. Fridrich, M. Goljan, D. Hogea, New methodology for breaking steganographic techniques for JPEGs, in: E.J. Delp III, P.W. Wong (Eds.), Proceedings of SPIE Security and Watermarking of Multimedia Contents IV, 2003, June, pp. 143–155. Santa Clara, CA, January 2003. Retrieved April 19, 2010, from http://www.ws.binghamton.edu/fridrich/Research/jpeg01.pdf

[50] H. Farid, S. Lyu, Higher-order wavelet statistics and their application to digital forensics, IEEE Workshop on Statistical Analysis in Computer Vision, 2003, June. Madison, WI. Retrieved April 19, 2010, from http://www.cs.dartmouth.edu/~farid/publications/sacv03.pdf

[51] D. McCullagh, Secret Messages Come In .Wavs, WIRED News 2001, February 20. Retrieved April 15, 2010, from http://www.wired.com/news/politics/0,1283,41861,00.html

[52] P. Hayati, V. Potdar, E. Chang, A survey of steganographic and steganalytic tools for the digital forensic investigator, in: Proceedings of the Workshop of Information Hiding and Digital Watermarking, 2007, July Moncton, New Brunswick, Canada.

[53] WetStone Technologies, Gargoyle Investigator, Retrieved April 20, 2010, from http://www.wetstonetech.com/cgi-bin/shop.cgi?view,2, 2010.

[54] WetStone Technologies, Stego Suite, Retrieved April 20, 2010, from http://www.wetstonetech.com/cgi-bin/shop.cgi?view,1, 2010.

[55] Steganography Analysis and Research Center (SARC), StegAlyzerAS. Retrieved April 20, 2010, from http://www.sarc-wv.com/products/stegalyzeras.aspx, 2010.

[56] OutGuess, Steganography Detection with Stegdetect. Retrieved April 20, 2010, from http://www.outguess.org/detection.php, 2004.

[57] Steganography Analysis and Research Center (SARC), StegAlyzerSS. Retrieved April 20, 2010, from http://www.sarc-wv.com/products/stegalyzerss.aspx, 2010.

[58] J.R.H. Martin, M. Kutter, Information retrieval in digital watermarking, IEEE Commun 39 (8) (2001, August) 110–116.

[59] S. Voloshynovskiy, S. Pereira, T. Pun, J.J. Eggers, J.K. Su, Attacks on digital watermarks: classification, estimation-based attacks, and benchmarks, IEEE Commun. 39 (8) (2001, August) 118–126.

[60] F.A.P. Petitcolas, 'mosaïc' attack, Retrieved April 20, 2010, from http://www.petitcolas.net/fabien/watermarking/2mosaic/index.html, 2009, June 20.

[61] J. Kelly, Terror groups hide behind Web encryption, USA Today Online 2001, February 5. Retrieved April 20, 2010, from http://www.usatoday.com/tech/news/2001-02-05-binladen.htm

[62] G. Kolata, Veiled messages of terror may lurk in cyberspace, The New York Times Online 2001, October 30.Retrieved April 20, 2010, from http://www.nytimes.com/2001/10/30/science/physical/30STEG.html?pagewanted=1.

[63] F. Manoo, The Case of the Missing Code, Salon.com. Retrieved April 20, 2010, from http://www.salon.com/tech/feature/2002/07/17/steganography/, 2002, July 17.

[64] D. Montgomery, Arrests of alleged spies draws attention to long obscure field of steganography, The Washington Post 2010, June 30. Retrieved September 6, 2010, from http://www.washingtonpost.com/wp-dyn/content/article/2010/06/30/AR2010063003108.html

[65] N. Shactman, FBI: spies hid secret messages on public Websites, WIRED Magazine Online 2010, June 29. Retrieved September 6, 2010, from http://www.wired.com/dangerroom/2010/06/alleged-spies-hid-secret-messages-on-public-websites/

[66] J. Callinan, D. Kemick, Detecting Steganographic Content in Images Found on the Internet, Department of Business Management, University of Pittsburgh at Bradford, 2001. Retrieved April 20, 2010, from http://www.chromesplash.com/jcallinan.com/publications/steg.pdf

[67] B. Nelson, A. Phillips, F. Enfinger, C. Steuart, Guide to Computer Forensics and Investigations, third ed., Thomson Course Technology, Boston, 2008.

[68] U.S. Department of Justice, Forensic Examination of Digital Evidence: A Guide for Law Enforcement. Office of Justice Programs, National Institute of Justice, 2004, April. NCJ 199408. Retrieved April 20, 2010, from http://www.ncjrs.org/pdffiles1/nij/199408.pdf

[69] U.S. Department of Justice, Searching and Seizing Computers and Obtaining Electronic Evidence in Criminal Investigations. Criminal Division, Computer Crime and Intellectual Property Section, 2009. Retrieved April 20, 2010, from http://www.cybercrime.gov/ssmanual/ssmanual2009.pdf

[70] J. Seward, The debtor's digital reckonings, Int. J. Digital Evidence 2 (2) 2003, Fall. Retrieved April 17, 2010, from http://www.utica.edu/academic/institutes/ecii/publications/articles/A0AE0A62-D73A-CB30-B0C1363FC9B1E185.pdf

About the Author

Gary C. Kessler is the president of Gary Kessler Associates, a digital forensics and information security consulting and training firm in Burlington, Vermont. Gary holds a B.A. in Mathematics, an M.S. in Computer Science, and a Ph.D. in Computing Technology in Education. He is a Certified Computer Examiner (CCE) and Certified Information Systems Security Professional (CISSP). Gary is also an adjunct associate professor at Edith Cowan University in Perth, Western Australia and a member of the Vermont Internet Crimes Against Children (ICAC) Task Force. In addition, he is the editor in-chief of the *Journal of Digital Forensics, Security and Law*. His e-mail address is *gck@garykessler.net*.

Chet Hosmer is Senior Vice President, Chief Scientist, and co-founder of WetStone Technologies, Inc., now a subsidiary of Allen Corporation of America. Chet, currently located in Conway, South Carolina, received his B.S. in Computer Science from Syracuse University and Utica College where he now serves as a visiting professor. Chet serves on multiple digital forensic editorial boards and speaks around the globe on steganography, malware and live investigation methodologies, and a plethora of other cyber security related issues. His e-mail address is *chet@wetstonetech.com*.