Digital Steganography—An Introduction to Techniques and Tools

Michael Brian Pope
Department of Management & Information Systems, Mississippi State University, m pope5678@gmail.com

Merrill Warkentin
Department of Management & Information Systems, Mississippi State University

Ernst Bekkering
Department of Information Systems, Northeastern State University

Mark B. Schmidt
Department of Information Systems, St. Cloud State University

Follow this and additional works at: https://aisel.aisnet.org/cais

Recommended Citation
Pope, Michael Brian; Warkentin, Merrill; Bekkering, Ernst; and Schmidt, Mark B. (2012) "Digital Steganography—An Introduction to Techniques and Tools," Communications of the Association for Information Systems: Vol. 30, Article 22.
DOI: 10.17705/1CAIS.03022
Available at: https://aisel.aisnet.org/cais/vol30/iss1/22

This material is brought to you by the AIS Journals at AIS Electronic Library (AISeL). It has been accepted for inclusion in Communications of the Association for Information Systems by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.
Steganography is the art and science of hiding information. In the digital realm, steganography (which literally means “covered writing”), involves hiding data or messages in digital files and other digital structures. The carriers holding the hidden content may appear to be innocuous, and would be ignored by a casual observer. The field of digital information hiding has grown significantly since the 1990s. Evidence of this growth can be seen at workshops on information hiding and in occasional reports of use by criminals and terrorists as reported in the popular press. In contrast to cryptography where the message is encoded, the purpose of steganography is to hide the fact that a message is being sent. Once encoded, a cryptographically altered message typically appears unrecognizable and would raise suspicions. The primary advantage of steganography over cryptography is that the carriers do not attract attention to themselves, to messengers, or to recipients. Modern information technology enables novice computer users to create steganographic messages, transmit, and unhide them without special expertise. This article presents an overview of steganography’s history and the categories of steg methods, followed by a discussion of the application areas for steganography. We also present some technical details of the techniques and software for applying steganography, including some security-related attack issues. Our article concludes with a presentation of some key topics for the reader’s consideration.

Keywords: steganography, steganalysis, encryption, hidden message, security, computer crime.
I. INTRODUCTION
Two interesting examples of steganographic applications were presented in the media in early in August 2011: one of these applications as a potential benefit, and another a potential threat. The two applications were named “Telex” and “Stegobot.” In the first case, researchers at the University of Michigan used steganography to bypass censorship by hiding redirection information in a message to a non-blocked website, which would forward the transmission to a blocked website [Krebs, 2011]. In the second case, researchers from the University of Illinois at Urbana–Champaign and the Indraprastha Institute of Information Technology in New Delhi, India, designed a proof-of-concept botnet that used the popular Facebook website to steal and hide private data in pictures on Facebook [Liebowitz, 2011].

In “A Few Words on Secret Writing,” Edgar Allan Poe [1841] writes, “As we can scarcely imagine a time when there did not exist a necessity, or at least a desire, of transmitting information from one individual to another, in such manner as to elude general comprehension; so we may well suppose the practice of writing in cipher to be of great antiquity.” Throughout history, protecting information has been a vital function in many contexts. Although cryptography—the transformation of messages so they can be decoded only by recipients who have the decryption key—has received a great deal of attention in the era of computers, other methods of securing information exist that may be just as important. One of these is steganography, the study of hiding messages and other data. Literally meaning “covered writing,” steganography can often pass through checks that would intercept encrypted messages due to their suspicious appearance. Cryptographically altered messages typically appear unrecognizable and raise suspicion, but the “cover” in steganography does not attract attention to senders, messengers, and recipients alike [Warkentin et al., 2008].

Essentially, the idea behind steganography is deception and security through obscurity. A hidden message is injected into a “carrier” medium or carrier message. The carrier, and not the hidden message included, appears to be the relevant item. Ideally, the carrier with the hidden content is indistinguishable from the carrier prior to injection, at least it appears so to third parties inspecting it. Of course, some change of the carrier is inevitable after the hidden message has been included. This change in carrier can be used to discover hidden content, and the success of the technique depends on a combination of the ability to hide content and reduction of change in original carriers. If a steganographic technique is successful, attempts to intercept the message should fail to separate innocuous items from items with hidden content by parties not among the intended recipients. Strong steganographic techniques combine high hiding ability with low probability of detection.

Many people have used steganography without realizing it. For instance, a simple method of steganography is writing a letter on the back of a photograph, and hiding the message by inserting the photograph into a frame. Even when crossing borders at a government checkpoint, the message would likely pass customs inspectors checking travel documents unless a full search was conducted. In the digital age, steganography can take many forms, including hiding data in image files, multimedia files, and documents. Even the use of computer viruses has been suggested to hide the presence of steganography [Hansmann, 1997]. Basic types of steganography will be discussed later.

II. HISTORY
Antiquity
Steganography is far from new, and its initial simplicity has grown in sophistication since its inception. Just as cryptography dates back to antiquity and specifically Julius Caesar, recorded uses of steganography go back to ancient Greece. The historian Herodotus reported in about 474 B.C. how Histaeus of Miletus concealed messages by tattooing them on the shaved scalp of slaves and waiting until regrown hair hid them [Kahn, 1996]. Less invasive forms of steganography soon appeared. The Greek soldier Demeratus inscribed a message that King Xerxes planned to invade Greece on the wood under the wax on a writing tablet. Mathematicians in China and Italy created similar techniques independently [Katzenbeisser and Petitcolas, 2000].

Middle Ages to Victorian Age
In the Middle Ages, two authors wrote seminal works on steganography. Johannes Trithemius (1462–1526) wrote the three volumes of Steganographia (ca. 1499) which superficially describes black magic but contains treatises on cryptography and steganography hidden by simple substitution methods. More than a century later, Gaspari Schotti...
published Steganographyica [1665] which focuses on techniques with text, invisible inks, and messages hidden in music. In the nineteenth century, the new telegraph significantly increased information transmission speeds. Almost immediately, businesses and individuals tried to conceal some message content with various steganographic disguises [Standage, 1999]. Late in the same century, Lord Baden-Powell worked as a scout for the British army and hid drawings of positions of Boer artillery bases in drawings of butterflies.

Twentieth Century and Beyond
As late as World War II, spies and resistance fighters wrote messages with invisible ink (juices, urine, or milk) and revealed the message by heating the document. The invention of microfilm allowed hiding microscopic images under fingernails in the Russian war of 1905 and the use of microdots in World War I [White, 1989]. The advent of computers, and especially the development of the Internet, has moved steganography into the digital realm. International workshops on information hiding and steganography have been held regularly since 1996 [Moulin and O'Sullivan, 2003]. In the first International Workshop on Information Hiding, participants defined the following terminology in steganography. The embedded data is the information to be hidden in the cover: the original, innocent file such as an image, audio, text, or video. The process itself is labeled embedding, and the cover and embedded data together form the stego data [Pfitzmann, 1996]. These definitions are still valid, with the caveat that the variety of carriers has increased. As we will discuss later, entire file systems can be used as steganographic systems. Since the first workshops on information hiding in the 1990s, the majority of development and use of computerized steganography has occurred since 2000 [Cole, 2003]. Steganography can hide in Internet telephony systems such as Skype [Mazurczyk and Szczypiorski, 2008]. Purdue University research found evidence of criminals using steganography tools, mainly in child pornography and financial fraud [Higgins, 2007]. In 2010, Russian spies used steganography software developed by the Russian intelligence service SVR to communicate with each other and their agencies [Higgins, 2010].

III. DEFINITION AND TAXONOMY OF STEGANOGRAPHY
Johnson [1995b] defines steganography as “the art of concealing the existence of information within seemingly innocent carriers.” Kessler [2004] provides a taxonomy of steganographic techniques (Figure 1). Linguistic steganography uses language to hide messages in symbols or signs (visual semagrams) or the appearance of text through font or spacing (text semagrams). Open codes hide the presence of messages in ways not obvious to casual observers and can use language meaningless to others (jargon code), special rules (null cipher), or physical templates (grille cipher). Technical steganography uses scientific methods, such as invisible inks or microdots. In information systems, digital steganography spans this classification scheme, from hiding messages in slightly altered images (technical steganography) to using simulated spam (grille cipher).

![Figure 1. Taxonomy of Steganography Techniques [Kessler, 2004]](image)

Anyone with information to conceal should understand steganography. Military services, intelligence services (e.g., Reporter, 2004), and criminals (e.g., Ringelesltn, 2004) alike have used it, and new concerns have surfaced with the growth of modern terrorist organizations [CNN, 2012; Gallagher, 2012]. However, businesses must also consider steganography for safeguarding sensitive data. James Wingate, director of the steganography analysis and research center at Backbone Security, states that corporate insiders could use steganography to steal sensitive data or intellectual property and sneak them out hidden in images or other files. “Over time and as [law enforcement] countermeasures get better … [criminals] will naturally be forced to migrate to more technically sophisticated information-hiding techniques,” he says. “If it’s there, they will use it” [Higgins, 2010]. With this in mind, security professionals should understand how steganography works, how it is used, and how to develop defenses against it.
IV. APPLICATION AREAS

Steganography, as a technique to keep information concealed, has its most obvious uses in military applications, illegal activities, and theoretical mathematical aspects dealing with decoding or finding information. Additionally, the technology has many applications for legitimate business and personal purposes, if properly applied. User-friendly software and interfaces make steganography accessible to ordinary users with minimal technical training or theoretical preparation, often requiring little more than basic computer skills [Bartlett, 2002]. Examples of tools and tutorials will be discussed in later sections.

One use is to help keep confidential files private. This is important in situations where, for instance, an employee of an organization carries a laptop with sensitive data while traveling. Cryptographic techniques would signal the presence of data protection. Furthermore, “cold boot attacks” use power disruptions and subtle design aspects of RAM modules to detect encryption keys and defeat some cryptography [Halderman et al., 2009]. Code breakers can use similar attacks against other areas of memory. For example, the operating system may not clear the swap file for the virtual memory system after power disruptions. Steganography adds an additional layer of protection. It obfuscates the location and nature of vulnerable data, thereby rendering the use of decryption moot or ineffective. In times of political unrest, corporations could use steganography to safeguard corporate data. In some countries, government actions may adversely affect business, including attempts to nationalize corporate assets. Obviously, in such cases, leaving sensitive information vulnerable is undesirable. Steganography can hide the presence of the data in local storage, and it can transfer data out of the country safely. Both would protect sensitive customer information from foreign governments, but users need to consider the legality of these uses.

A third use of steganography involves non-repudiation [Maxemchuk, 1994]. End users and corporations alike can embed information in digital files that uniquely identifies the creator, location and time of creation, and other relevant information. Television broadcasts can protect data by encoding within the audiovisual stream itself.

Other applications of concealing data, of varying levels of complexity and sophistication, exist and can be readily applied. Organizations increasingly consider steganography as a useful business tool, and the number of carriers is increasing. For example, database developers use steganography to place identifiers in database relationships [Agrawal et al., 2003]. In general, steganography can be used either independently or in combination with other technologies. It is a powerful approach to prevent unwanted detection of messages, simply by denying third parties knowledge of the presence or location of protected information.

V. PRIMER ON STEGANOGRAPHIC TECHNIQUES

The considerable research interest in steganography suggests that the details of steganographic communication are more complicated than they might first appear. In order to illustrate what steganography is, it is useful to illustrate what it is not [Wang and Wang, 2004].

First, cryptography allows two computers to securely send and receive messages by scrambling the message itself. Data is scrambled in a reversible manner by utilizing “keys” to encode and decode it. Some encryption algorithms are symmetric (single-key encryption), while others are asymmetric and use separate keys for senders and recipients (public key encryption). As Johnson [1995a] states: “an encrypted message may draw suspicion while a hidden message will not.” Consequently, encrypted messages are often hidden with steganography. In this way data is both hidden and encoded.

Watermarking is another technique often associated with steganography. Like steganography, watermarking embeds messages within messages [Collberg and Thomborson, 1998]. Some watermarks are intentionally visible, as in the case of television broadcasts including a station icon. This disqualifies visible watermarks from steganography as “hidden writing.” Other watermarks are hidden to indicate ownership or source. An example is the distribution of music files provided without Digital Rights Management (DRM). Each file may be watermarked with unique identifiers at the time of purchase and download. The original purchaser of the file can be traced if the music file is distributed on file sharing services or by other means, which discourages unauthorized copying. However, whereas hidden stego messages are intended to be received and read each time, hidden watermarks are used only incidentally upon infration. Another key difference is the permanence of the hidden message. Stego messages lose their value as information becomes outdated and stale, and hidden watermarks are intended to be permanent. A key aspect of watermarking is the durability of the mark [Swanson et al., 1996]. Attempts to remove the watermark should either fail to destroy the identifier or render the carrier useless. In the past, “lossy” data compression techniques, such as converting images to JPEG format, often altered or destroyed watermarks. Techniques have now been developed to address the difficulties encountered in encoding information within this type of data [Morkel et al., 2005]. Table 1 summarizes the differences between the steganography, cryptography, and watermarking.
Table 1: Comparison of Steganography, Cryptography, and Watermarking

<table>
<thead>
<tr>
<th>Technique</th>
<th>Purpose</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steganography</td>
<td>Hiding existence of digital content from outsiders</td>
<td>Content generally of limited time value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Needs carrier file</td>
</tr>
<tr>
<td>Cryptography</td>
<td>Rendering the digital content inaccessible to outsiders</td>
<td>Content generally of limited time value</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No need for carrier file</td>
</tr>
<tr>
<td>Watermarking</td>
<td>Protection of digital content of carrier</td>
<td>May or may not be readily detectable</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Durability is essential</td>
</tr>
</tbody>
</table>

Steganography techniques have different levels of concealment and different insertion techniques [Warkentin et al., 2008]. Steganographic content can be inserted with multiple techniques: injection, substitution, and file creation. These three forms of pure steganography differ in level of change to the carrier file, and rely only on hiding the message. The content is not protected when the presence of a message is discovered. An additional step can be the exchange of a key, similar to encryption keys. Without the key, the steganographic algorithm is still not able to extract the message.

Injection techniques combine the carrier and the hidden content in a single file without altering the integrity of either part. For instance, data files often include some space deliberately left empty to allow compatibility between software versions. Future versions can use the blank space in older files to store additional features; older software versions simply ignore the reserved bits and leave them unchanged. The overall format is left undisturbed. Injection of hidden content in reserved space is easy to implement, but also easy to detect since the location of empty file space is commonly known. Examples include using the “hidden” tag in Web pages, storing data in unused space in file headers, data packets sent over networks, and unused disk space [Johnson et al., 2001]. Data hiding capacity is limited, and modern techniques usually involve some modification of the carrier file.

Substitution techniques involve actual modifications to the message within the carrier file. The hidden message becomes a direct part of the data file, which can still be processed by regular application software. Only special steganographic software can extract the message, if necessary with the special key. Substitution algorithms must be subtle, lest they damage the message of the carrier file. When damaged, the carrier may be obviously altered or cannot be used by regular application software. Figure 2 presents a simple form of substitution in an ASCII text file. In this example, we use the 3-bit unsigned integer 5 as our message encoded in a text file. The number 5 encodes to 101 in binary notation. The ASCII chart shows that the conventional space character is hexadecimal 20, but the visible character for hexadecimal FF on the extended ASCII chart used by most text editors also shows as a blank. We can take advantage of this to encode the binary string 101 in the spaces of the file. Each space in the file represents a single bit of our message—hexadecimal 20 for a 0, and hexadecimal FF for a 1. As Figure 2 indicates, a typical text editor does not notice this change. It displays both the strings encoded with hexadecimal FF and hexadecimal 20 as blank spaces. The effect of the technique is similar to the encoding scheme found in some steganography software such as wbStego (http://wbstego.wbailer.com/).

Substitution tends to work best with more complicated data, where each individual bit has less impact on the overall data being conveyed. This prevents noticeable changes in the data. For instance, modification of 8-bit images is less versatile to work with than 24-bit, or “true color,” images [Johnson and Jajodia, 1998]. The larger file size of 24-bit...
images provides more space to hide content, and the use of some 8-bit images requires modifications to the color palette. Due to the larger number of colors (16 million vs. 256) in 24-bit images, color changes are more subtle and less obvious. Kessler [2001] presents an example using the least significant bits to carry the data in a carrier image file. Typical 24-bit images are encoded with 8 bits each for red, green, and blue. Values range from 0 (total absence of the color) to 255 (maximum saturation of the color). Color values are typically expressed as three-element tuples in RRR GGG BBB format where each trio is a base-10 number encoded as 0–255. The typical human eye will not notice differences in color changes of a single level. The same integer 5 (binary 101) change as in the text file example barely changes a single yellow pixel. Yellow is a mix of red and green in monitors, unlike the use of yellow only with the primary colors (red, blue, yellow) system. Encoding maximum strength yellow results in (255, 255, 0) or binary:

```
11111111 11111111 00000000
```

Hiding the binary string 101 in the last bit of each octet—the least significant bit (LSB)—produces:

```
11111110 11111111 00000001
```

Note the minute change to the color: only a little red was lost and a little blue added. Green remained the same, and three bits of the message only required changing two bits in the carrier. The net visual change is likely unnoticeable to the casual observer, and the general integrity of the image is preserved. For every pixel in a 24-bit image, three bits can easily be hidden with this technique. The fact that only roughly 50 percent of the least significant bits will be changed further adds to the unlikely hood of this changes being detected by the human eye (a 0 or 1 will be replaced by a 0 or a 1, so on average, only about half will actually be changed). Similar LSB techniques use other types of media, such as audio- or video-files. A number of other techniques, such as fractal-based algorithms [Davern and Scott, 1996], use image files as well.

A third insertion technique is file creation where a new file is created to act as carrier. An excellent example is SpamMimic (http://www.spammimic.com). Users of the website can create messages simulating unsolicited e-mail advertisements or “spam” messages. To the casual observer, the new carrier file appears as a spam message, but the hidden message is hidden within the words. A SpamMimic user can generate the carrier message on the website and copy it to an e-mail client and send it to a recipient; the recipient then copies the carrier message and pastes it to the website to extract the hidden content. The technique is somewhat inefficient since each message requires multiple non-automated steps and hundreds of words are needed to hide a simple phrase.

Table 2 presents a summary of insertion techniques.

<table>
<thead>
<tr>
<th>Technique</th>
<th>Method</th>
<th>Effect on Carrier File</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Injection techniques</td>
<td>Using built-in information recording tools or “open” file space</td>
<td>No change of content</td>
<td>Very limited hiding capacity</td>
</tr>
<tr>
<td>Substitution techniques</td>
<td>Part of digital content of carrier file changed to reflect stego message</td>
<td>Some degradation of content quality</td>
<td>Increased risk of detection with increasing volume of stego content</td>
</tr>
<tr>
<td>File creation</td>
<td>Stego message hidden in larger amount of new, irrelevant digital content</td>
<td>None—new carrier file created</td>
<td>Inefficient, detection risk highly dependent on context of message</td>
</tr>
</tbody>
</table>

All three techniques can be combined with encryption. The stego content is encrypted as it is included in the carrier file, and the effect on the carrier is the same as for pure steganography. The only significant difference lies in detection risk caused by key exchange.

### Audio Files

In nature, sound consists of analog waves with different frequencies and different amplitudes. Digitally, sound is converted with pulse-code modulation where waves are sampled at regular intervals and each sample represented in discrete values (Figure 3). Together, the sampling rate and bit depth (number of values for strength) determine the fidelity to the analog signal. Another factor in digitization of audio is the use of compression codecs. These compression algorithms allow more efficient transmission and storage of audio without or with little loss (lossless vs. lossy compression). Steganography in audio can be used to hide information in the frequency, amplitude, phase, files spaces, or compression components.
As in text steganography, LSB encoding can hide the information in the least significant bit. For instance, in 16-bit audio, only the 16th bit of each sample is changed. This minimally alters the amplitude of the samples, but may not be detectable to the human ear, especially in higher levels of bit depth. Dutta et al. [2009] review other methods in uncompressed signals. In parity coding, the signal is broken down in regions of samples instead of separate samples and the parity bit of regions adjusted to carry the message (Figure 4). In phase coding, the timing of the waves are slightly adjusted to store the message (Figure 5). In Spread Spectrum Coding, the message is embedded in the frequency spectrum of the sound file. This code is independent of the sound signal, and can be either embedded in specific parts of the frequency channel for the duration of the transmission (Direct Sequence Spread Spectrum), or the carrier switches rapidly among multiple frequencies in a pseudorandom sequence known to both sender and receiver (Frequency Hopping Spread Spectrum). Finally, in Echo Hiding, the message is carried in an echo to the original signal. The amplitude of the echo can be kept low to avoid detection, and the timing of the echo represents the binary zeros and ones (Figure 6).

Disadvantages of these insertion techniques relying on modification of analog and digital waves include the introduction of noise in the signal; but moreover, most transmissions of audio now occur in compressed format. The mere presence of audio transmission in uncompressed format suggests an unusual event. Consequently, steganography techniques have focused on embedding information in MP3 files, the most popular audio compression format. The MP3 format is a “lossy” format, where some signal is lost during the compression process. To avoid data loss, insertion of the hidden message (in plain form or encrypted form) occurs after the filtering and transformation stages of MP3 encoding (Figure 7).

![Figure 4. Parity Coding [Dutta et al., 2009]]
Image Files and Video Files
Like audio files, images and video offer greater storage capacity for hidden data. On the Internet, the most popular image formats are the Graphics Interchange Format (GIF), Joint Photographic Experts Group (JPEG or JPG), and Portable Networks Graphics (PNG). Older steganography methods involving graphics mostly use the Bitmap (BMP) format, and newer methods use the other three file types listed. Cheddad et al. [2010] discuss image-format, spatial domain, and frequency domain techniques.
In image-format steganography, data is hidden either in meta-data of the image or in a block attached to the end of the file. Image files do not consist only of data about color, intensity, and location of image blocks (pixels). Depending on the file format, additional data is included about the image itself (metadata). One standard is EXIF, and editors such as EXIF Maker demonstrate the potential for using metadata to store messages (Figure 8).

An even cruder method is attaching data to the image file after the End of File (EOF) marker (Figure 9). This can be done with simple shell commands like:

```
C: > copy cover.jpg / b + hidden.txt / b stego.jpg (Windows)     or     cat cover.jpg hidden.txt > stego.jpg (Linux)
```

Since the image software stops processing the image at the EOF marker, the hidden data does not affect display of the image. However, it can readily be detected with basic software such as NotePad.

In the spatial domain, data is directly stored in information about the pixels. One example is the previously mentioned LSB encoding where the last bit in each of the three color components is used, another example is LSB palette encoding where the information is stored in an index of colors of the GIF image. An image of the palette...
demonstrates that colors are represented in order of frequency. This allows minor LSB adjustments without noticeable changes.

These techniques are relatively simple and have resulted in a multitude of steganography applications, but cannot be used in files which utilize compressed formats. Since most image files transferred over networks are either small (with low payload capacity) or compressed (which can destroy the hidden data), frequency domain steganography is preferable.

In the frequency domain, images are transformed (compressed) before the message is embedded. Common compression algorithms are the Fourier Transform (FT), Discrete Cosine Transform (DCT), and Discrete Wavelet Transform (DWT). In compression of JPEG image files, the most popular image file format, compression moves through several stages.

Color transformation, Down-sampling, DCT, and Quantization all form part of the “lossy” stage of JPEG compression. However, the Huffman Encoding algorithm is lossless, and the results of quantization can be slightly modified before the final phase [Morkel et al., 2005]. Other techniques in the frequency domain use similar techniques making small changes in the results of the compression transformations. Comparing the spatial and frequency domains, Morkel [2005] suggests that spatial domain techniques may have a better hiding capacity, but frequency domain techniques may be safer as transport vehicles.
Hiding data in digital files is also limited due to changes in the carrier file, which can lead to discovery of the presence of steganography content. Larger amounts of data can be hidden by hiding content files themselves. Depending on the context and the amount of data to be hidden, different tools and techniques can be used. Some techniques do not require any special tools because they have been provided by the operating system. File deletion, for instance, can be used as a simple form of steganography. Most operating systems, when instructed to delete a file, do not actually erase file contents; rather, the entry in the file directory is removed and the space is marked as “clear” [Garfinkel and Shelat, 2003]. This is primarily for performance reasons, since marking storage space as unused requires less activity than actually deleting the data from disk. In this so-called “slack space,” old data remains until overwritten by new data. This creates the ability to hide files by erasing them from portable media, such as a flash drive. The user could transport the innocuous-looking disk, and use an undelete program run on it to recover the data. In transit, users would risk data destruction due to accidentally accessing the disk and writing to it, as the content is contained in blocks considered “blank” by the operating system. Even if the operating system does not support the recovery of data with built-in utilities or functions, hexadecimal editors or third-party tools can examine the exact structure of files and file system entries on the disk. “Unformat” programs can use these techniques at disk level. Simulated formatting by removing the File Allocation Table makes the disk appear blank, unused, and possibly entirely unusable. These approaches are similar to those discussed in Kipper [2004]. More sophisticated tactics might involve discarding standards format entirely and writing directly to the medium. This is the subject of the next section.

VI. PORTABLE FILE-SYSTEM STEGANOGRAPHY

When the amount of data to be hidden is large, hiding entire files, folders, or file systems may be more effective than hiding data inside carrier files. Steganographic file systems exist for the Linux operating system [McDonald and Kuhn, 2000], though they are niche applications, and are not considered standard by the Linux community. The best known, StegFS, has in fact not been maintained for multiple years and is intended to work with older Linux kernels [McDonald, 2004]. Though theoretically usable for steganography, these systems are not very popular. Less invasive methods involve using the file system itself to hold the data [Kipper, 2004] and create virtual file systems. File system steganography can be combined with encryption. This adds an extra level of protection and decreases the risk of discovery of hidden data if the presence of the data is detected. Finally, file systems should be transportable between different operating systems to prevent compatibility problems. Ideally, these features would be combined into a single program.

An example of programs including these features is TrueCrypt (http://www.truecrypt.org), an open source program for Windows, Linux, and Mac OS X. It creates a file, or “volume,” that simulates an encrypted virtualized disk drive. Just as applications like MagicISO (http://www.magiciso.com) mount ISO files as virtualized drives, operating systems can connect to TrueCrypt volumes and treat them virtual disk drives. The virtual drive appears complete with its own letter, mount point, or analogous interface structure depending on the operating system. While TrueCrypt specializes in encryption, it includes a steganography function through the ability to create a so-called “hidden volume.” The program creates two TrueCrypt volumes within the same file. The concept is illustrated in Figure 12. The outer drive is the deceptive drive and uses the beginning of the file, much like a regular TrueCrypt volume. The hidden drive resides at the end of the file, simulating the end of the drive space. Two keys are required for accessing the TrueCrypt volumes. The first key mounts the outer TrueCrypt drive, and the second key opens the hidden part. With only the first key, the TrueCrypt drive shows decoy files and shows the “empty space” with random data. The header is not accessed, and as it is scrambled whether or not there is a hidden drive, it is impossible to prove that a hidden drive exists without the password. The second drive is, therefore, a “hidden message” in the form of an entire virtual disk drive that appears as random data on the end of the deceptive volume, unless it is made visible with the second key. TrueCrypt volumes emulate drives that use typical file systems such as FAT and NTFS. Regular disk utilities can operate on the volumes, and secure deletion programs can run on the blank portion or any other part of the drive. This might result in random data overwriting part of the hidden volume. Another weakness is the extreme care which must be exercised to avoid destroying the hidden data by writing too much data

<table>
<thead>
<tr>
<th>Table 3: Comparison of Spatial and Frequency Techniques, Adapted from Morkel et al., 2005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invisibility</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>LSB in BMP (spatial)</td>
</tr>
<tr>
<td>High *</td>
</tr>
<tr>
<td>High</td>
</tr>
<tr>
<td>Low</td>
</tr>
<tr>
<td>Low</td>
</tr>
<tr>
<td>Low</td>
</tr>
<tr>
<td>Low</td>
</tr>
</tbody>
</table>

* depends on cover image used
to the deceptive drive and destroying the hidden drive. TrueCrypt includes a function to protect the hidden volume at mount time, but enabling this function indicates the presence of a hidden volume. Not using the protection function and not using the outer volume for actual work provides sufficient security of the hidden volume. Furthermore, overwriting the hidden volume is an asset when data must be quickly destroyed without detection. Despite the risk of accidental data loss, virtualized file systems offer large storage space, convenience, robustness, sophistication, and versatility.

**Figure 12. Configuration of Regular and Hidden TrueCrypt Volumes, Adapted from TrueCrypt, 2010**

In this section, we discussed the use of entire file systems to hide data and messages. In the next section, we present an overview of applications.

**VII. STEGANOGRAPHY TOOLS**

Hiding complete files, folders, and file systems typically require large amounts of storage capacity. However, storing small amounts of steganographically-encoded information within other files is easier and allows transportation of carrier and message over networks. File-system steganography is by its nature more suitable for use on local systems. Table 4 illustrates commonly used prepackaged tools and utilities, free and commercial alike, with proprietary and open-source options.

As the table shows, steganography can be used with a variety of file types, and many tools can be used without much instruction. For instance, wbStego includes both a wizard mode and a diagram mode, allowing users to be led step by step through the encoding and decoding processes. More specialized tools focus primarily on only one file type. OpenStego (http://www.openstego.info/ or http://sourceforge.net/projects/openstego/) is an example of an open-source program. It is designed to interoperate with alternative algorithms as plugins. By default, it focuses on creating image files. Written in Java, it is also platform-independent as long as an appropriate Java Run-time Environment (JRE) is used. Like wbStego, OpenStego can be used on multiple operating systems. Though it can be used from the command line, less experienced users can use a simple graphical user interface (Figure 13). In the GUI, users select the message file (hidden message), cover file (carrier file), and output file, and click “OK.” The message can be compressed before insertion, can be password-protected with encryption, and can even be hidden in a randomly generated output image file. To decode the message, recipients select the “extract” tab, specify the carrier file and the desired name of the message file, and a password if necessary.

This section presented some examples of steganography tools. A more complete listing can be found at Dr. Neil Johnson’s website at http://www.jjtc.com/Steganography/tools.html. In the next section, we will discuss attacks on information systems using steganography.
Table 4: Commonly Used Steganography Tools, Adapted from Warkentin et al., 2007

<table>
<thead>
<tr>
<th>Software</th>
<th>Text</th>
<th>Audio</th>
<th>Images/Video</th>
<th>Files/Folders</th>
<th>Cost</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera/Shy</td>
<td></td>
<td>X</td>
<td>Free</td>
<td></td>
<td></td>
<td>Scans and decrypts stego content in Web pages</td>
</tr>
<tr>
<td>Camouflage</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Free</td>
<td></td>
<td>No longer supported or updated</td>
</tr>
<tr>
<td>Data Slash</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Free</td>
<td>$39.95</td>
<td></td>
</tr>
<tr>
<td>DriveCrypt</td>
<td>X</td>
<td></td>
<td>X</td>
<td>$69.95</td>
<td>Standard edition and “Plus Pack”</td>
<td></td>
</tr>
<tr>
<td>Folder Guard</td>
<td></td>
<td></td>
<td>X</td>
<td>$39.95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GifShuffle</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td></td>
<td>Last updated 2003</td>
</tr>
<tr>
<td>Hide in Picture</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hide N Seek</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Java-based</td>
<td></td>
</tr>
<tr>
<td>Hide4PGP</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Free</td>
<td></td>
<td>Also available for Linux</td>
</tr>
<tr>
<td>Invisible Secrets v4.0</td>
<td>X</td>
<td></td>
<td>X</td>
<td>$39.95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>iSteg</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Macintosh only</td>
<td></td>
</tr>
<tr>
<td>Magic Folders</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Max File Encryption</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Free</td>
<td>$49.95</td>
<td></td>
</tr>
<tr>
<td>Mosaig</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Web-based only</td>
<td></td>
</tr>
<tr>
<td>MP3Stego</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MP3Stegz</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSU StegoVideo</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MySecretFolder</td>
<td></td>
<td></td>
<td>X</td>
<td>$24.95</td>
<td>Free 30-day trial</td>
<td></td>
</tr>
<tr>
<td>OpenStego</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Java-based</td>
<td></td>
</tr>
<tr>
<td>OpenPuff</td>
<td>X</td>
<td>X</td>
<td>Free</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Our Secret</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>$24.95</td>
<td>Formerly: Steganography</td>
<td></td>
</tr>
<tr>
<td>Outguess</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Source code</td>
<td></td>
</tr>
<tr>
<td>Pict Encrypt 2.0</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Macintosh only</td>
<td></td>
</tr>
<tr>
<td>Puffer</td>
<td></td>
<td></td>
<td>X</td>
<td>$34.95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>QuickCrypto</td>
<td>X</td>
<td>X</td>
<td>£ 24.99</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>QuickStego</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecurEngine 2.0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Free</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SilentEye</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Also available on Mac and Linux</td>
<td></td>
</tr>
<tr>
<td>Snowdrop</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Still in beta</td>
<td></td>
</tr>
<tr>
<td>Spam Mimic</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Web-based only</td>
<td></td>
</tr>
<tr>
<td>Steganos Security</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>$69.95</td>
<td></td>
<td></td>
</tr>
<tr>
<td>StegFS</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Linux and NetBSD only</td>
<td></td>
</tr>
<tr>
<td>Steghide</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Linux only</td>
<td></td>
</tr>
<tr>
<td>StegoArchive</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>$21.95</td>
<td>multiple freeware and shareware programs on CD</td>
<td></td>
</tr>
<tr>
<td>S-Tools</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>No longer available from author, but can still be found on Internet</td>
<td></td>
</tr>
<tr>
<td>TrueCrypt</td>
<td></td>
<td></td>
<td>X</td>
<td>Free</td>
<td>Also available in Linux and Mac</td>
<td></td>
</tr>
<tr>
<td>wbStego</td>
<td>X</td>
<td>X</td>
<td>Free</td>
<td></td>
<td>Also available for Linux; last updated 2004</td>
<td></td>
</tr>
<tr>
<td>Xidie Security Suite</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>Free</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VIII. SYSTEM-LEVEL STEGANOGRAPHY ATTACK WITH COMPROMISED OPERATING SYSTEM AND DRIVER CODE

Even if computer users never feel the need to use steganography to covertly transmit messages, the technology may have relevance. For example, if a system has been compromised, intruders can attempt to covertly transfer information. Using an active administrator-level or system-level account to watch for desired data is likely to be detected, since active accounts are monitored by the operating system. First, detection is avoided by modifying the operating system kernel and/or device drivers. The kernel is the “core” of the operating system and typically loads software needed for interacting with physical devices on an as-needed basis. With most operating systems, driver usage is not actively monitored, and often the driver attains complete control. In some cases, a combination of driver and core modifications can even defeat cryptography built into the operating system through access to keys and encryption algorithms.
Once detection has been prevented, the intruder needs a “covert channel,” a hidden channel that transmits information between two programs [Lampson, 1973]. The use of data transfer abilities of the compromised machine, such as secretly installing an FTP server, can lead to detection due to suspicious network activity. Intruders can send (virtually) undetectable data transmissions by using network steganography. Rather than using dedicated transmissions, legitimate user transmissions include hidden content. Stolen data is transferred one piece at a time by exploiting nuances of network protocols. Since no unusual network activity is detected, and since the data transmission is part of regular activity of legitimate users, network steganography is difficult to detect and prevent.

The following example shows how network steganography uses the Transmission Control Protocol (TCP). TCP is part of the TCP/IP (Transmission Control Protocol/Internet Protocol) suite directing Internet data exchanges. Since ordinary Internet transmissions such as Web browsing and e-mail use TCP/IP, using these protocols will not raise suspicions if transfer of stolen data is part of network activity of authorized users. TCP is more complicated than other parts of the protocol suite in order to support features such as the “sliding window.” This feature determines if all parts of a transmission have been received and assembles them in proper order, even when received out of order. The packet header with instructions for processing transmitted data includes options which may not be used, as well as reserved places that were never activated to begin with. Figure 14 shows the header format of a TCP packet, including a reserved field (R1) and a two bits reserved as a flag field (R2). The padding between options and their storage space of multiples of four bytes offers further storage space.

Attackers can use a control file that keeps track of destination IP addresses and files to be transmitted. Modified code in kernel or drivers that assembles TCP packets reads the control file. Each new packet sent to a pre-specified IP address receives a new 4-bit chunk of the file to be transmitted. This continues until all data have been transmitted. Attackers can install other programs to deliberately trigger the transmission of illicit data to the desired address, such as using a command-line program to send data to a Web server run on the destination computer. Other command-line programs can destroy modified source files and configuration files. Finally, attackers would
have to modify the destination computers to strip the hidden data from the carrier transmission and to place the assembled files where they can be accessed. Camouflaging illicit activity on the destination computer is less critical, as long as it cannot be traced back to the attacker.

The technique in Figure 15 is similar to others previously discussed in the literature [Handel and Sandford, 1996; Katzenbeisser and Petitcolas, 2000]. It is somewhat simplified to make relevant pseudocode and illustration easier. While the example is relatively simplistic, the potential to create covert channels with empty spaces in network packet headers and network protocols is well-described in the literature. Systems administrators and researchers need to consider this type of attack, especially since more sophisticated attacks are possible [Rowland, 1997; Zander et al., 2007].

**IX. STEGANALYSIS**

Digital technology can be used not only to hide content, but also to detect and decode the same data. Like cryptography and cryptanalysis, steganography and steganalysis are two sides of the same coin. Research in steganography involves new techniques to hide and other techniques to detect and decode. Johnson [1995a] explains: “Steganography is the art of passing information in a manner that the very existence of the message is unknown. The goal of steganography is to avoid drawing suspicion to the transmission of a hidden message. If suspicion is raised, then this goal is defeated. Steganalysis is the art of discovering and rendering useless such covert messages.” This definition hints at the dual objectives of discovery and decoding.

Katzenbeisser and Petitcolas [2000] distinguish the following levels of steganalysis based on knowledge of the cover, the message, the algorithm, and the stego object (cover with message embedded):

- Stego only attack: Only the stego object is available for analysis.
- Known cover attack: The cover and the stego object are both available for comparison.
- Known message attack: The message is known and available for comparison with the stego object.
- Chosen stego attack: The stego object and the stego algorithm are available for analysis.
- Chosen message attack: Steganalysts use many steganographic tools for a chosen message and compare the results with the stego object to determine the algorithm.
- Known stego attack: The stego message, the stego tool (algorithm), and the cover message are all available for analysis.

In general, steganalysis results improve as more elements are known. Moving the analysis from detection only to detection and deciphering adds another level of complexity.

Methods to detect the presence of stego content include a variety of techniques. In the past, steganalysis tools usually detected only single applications. Later software uses more sophisticated methods like statistical Discriminant Analysis [Provos, 2004]. Special detection software based on the Steganography Application Fingerprint Database (SAFDB) can detect more than 800 stego applications [Backbone Security, 2010]. Likewise, the National Institute of Standards and Technology (NIST) includes the digital signatures of some stego software in the National Software Reference Library [NIST, 2009]. Other techniques involve the use of statistics of known properties. For instance, the presence of more than fifty near-identical colors in bitmap files suggests the use of least-significant bit techniques [Katzenbeisser and Petitcolas, 2000]. Finally, decoding of the message is not always necessary. Changing file formats, compression algorithms, and compression levels of graphical files can often destroy hidden content without noticeably affecting the integrity and function of the carrier.
X. SUMMARY

This tutorial provides a summary introduction to steganography. We have presented basic techniques, several useful tools and implementations, and illustrated several examples to encapsulate the general idea of hiding data in modern digital data exchange. Interested readers will find many sources of past and current research, and we encourage future development. New techniques will be invented, and old techniques will be used in new, innovative ways. With the increased emphasis on network security, military concerns, fighting crime, and terrorism, the need for a basic familiarity with these tactics and their potential use by and against organizations is obvious. Although it is difficult to put an exact dollar amount on the possible costs of the illicit use of steganography, it has the potential to create catastrophic losses when company secrets or national security are at stake. Interest in development and use of steganography should not be neglected, as illustrated by the response of James Goldman of Purdue’s Department of Computer and Information Technology in the 2010 case of Russian spies: “…. it would not surprise
me if these adversaries know that we have de-emphasized our interest in steg and therefore making it all the more appealing to them” [Higgins, 2010]. Even some experts have fallen into this trap: “Steganography becomes the focus of attention, dies down, and then the public is all over it again…. But it will never be pervasive, because the amount of data you can actually hide in the images is fairly small. And if someone wanted to steal intellectual property, it’d be easier to copy the data on a disk and carry it out in your pocket” [Niels Provos in Radcliff, 2002].

In closing, the authors would like to note that we do not advocate the use of these techniques or tools for any illegal or unethical activities. Rather, the examples here are to illustrate the concepts and increase awareness of important subject matter in the increasingly security-conscious, information-dependent, and information-centric society we live in today.

REFERENCES

Editor’s Note: The following reference list contains hyperlinks to World Wide Web pages. Readers who have the ability to access the Web directly from their word processor or are reading the article on the Web, can gain direct access to these linked references. Readers are warned, however, that:

1. These links existed as of the date of publication but are not guaranteed to be working thereafter.
2. The contents of Web pages may change over time. Where version information is provided in the References, different versions may not contain the information or the conclusions referenced.
3. The author(s) of the Web pages, not AIS, is (are) responsible for the accuracy of their content.
4. The author(s) of this article, not AIS, is (are) responsible for the accuracy of the URL and version information.


ABOUT THE AUTHORS

Michael Brian Pope is a doctoral candidate in Information Systems at Mississippi State University. He holds a bachelor’s and master’s degree in computer science from California State University, Sacramento. His current research interests include telecommunication, security, social media, knowledge management, and the legal aspects of IS. His work has been published in Communications of the Association for Information Systems and in the Proceedings of the Decision Sciences Institute annual conference, where it was nominated for best paper.

Merrill Warkentin is a Professor of Information Systems and the John and Carole Ferguson Notable Scholar at Mississippi State University. He earned his Ph.D. in MIS at the University of Nebraska–Lincoln. His research focuses on behavioral issues in IS security and on electronic group collaboration. He has authored over 250 manuscripts and six books. He is serving as Associate Editor for MIS Quarterly, European Journal of Information Systems, and Information & Management. He has chaired several international conferences, including IFIP and WISP. His work has been supported by the US Navy, NSA, the IRS, the UN, Homeland Security, IBM, and others. He has been a visiting scholar at over two dozen universities in eight nations, and has served as an ACM National Distinguished Lecturer. His work has appeared in MIS Quarterly, Decision Sciences, European Journal of Information Systems, Decision Support Systems, DATA BASE for Advances in Information Systems, Information Systems Journal, Communications of the Association for Information Systems, Communications of the ACM, and other journals and in numerous books.

Ernst Bekkering holds a BS in Physical Therapy in his native Holland and an MSIS and Ph.D. in Information Systems from Mississippi State University. He is an Associate Professor at Northeastern State University in Tahlequah, OK. He has co-authored journal articles and presented at conferences in the areas of educational assessment, using support applications in education, technology adoption, systems development risk, user perceptions, and information systems security.

EDITOR-IN-CHIEF
Ilze Zigurs
University of Nebraska at Omaha

AIS PUBLICATIONS COMMITTEE
Kalle Lytyinen
Vice President Publications
Case Western Reserve University
Ilze Zigurs
Editor, CAIS
University of Nebraska at Omaha
Shirley Gregor
Editor, JAIS
The Australian National University
Robert Zmud
AIS Region 1 Representative
University of Oklahoma
Phillip Ein-Dor
AIS Region 2 Representative
Tel-Aviv University
Bernard Tan
AIS Region 3 Representative
National University of Singapore

CAIS ADVISORY BOARD
Gordon Davis
University of Minnesota
Ken Kraemer
University of California at Irvine
M. Lynne Markus
Bentley University
Richard Mason
Southern Methodist University
Jay Nunamaker
University of Arizona
Henk Sol
University of Groningen
Ralph Sprague
University of Hawaii
Hugh J. Watson
University of Georgia

CAIS SENIOR EDITORS
Steve Alter
University of San Francisco
Michel Avital
Copenhagen Business School
Jane Fedorowicz
Bentley University
Jerry Luftman
Stevens Institute of Technology

CAIS EDITORIAL BOARD
Monica Adya
Marquette University
Dinesh Batra
Florida International University
Indranil Bose
Indian Institute of Management Calcutta
Thomas Case
Georgia Southern University
Evan Duggan
University of the West Indies
Andrew Gemino
Simon Fraser University
Matt Germonprez
University of Wisconsin-
Eau Claire
Mary Granger
George Washington University
Ake Gronlund
University of Umea
Douglas Havelka
Miami University
K.D. Joshi
Washington State University
Michel Kalika
University of Paris Dauphine
Karlheinz Kautz
Copenhagen Business School
Julie Kendall
Rutgers University
Nelson King
American University of Beirut
Hope Koch
Baylor University
Nancy Lankton
Marshall University
Claudia Loebbecke
University of Cologne
Paul Benjamin Lowry
City University of Hong Kong
Don McCubbrey
University of Denver
Fred Niederman
St. Louis University
Shan Ling Pan
National University of Singapore
Katia Passerini
New Jersey Institute of Technology
Jan Recker
Queensland University of Technology
Jackie Rees
Purdue University
Raj Sharman
State University of New York at Buffalo
Mikko Siponen
University of Oulu
Thompson Teo
National University of Singapore
Chelley Vician
University of St. Thomas
Padmal Vitharana
Syracuse University
Rolf Wigand
University of Arkansas, Little Rock
Fons Wijnhoven
University of Twente
Vance Wilson
Worcester Polytechnic Institute
Yajiong Xue
East Carolina University

DEPARTMENTS
Information Systems and Healthcare
Editor: Vance Wilson
Information Technology and Systems
Editors: Dinesh Batra and Andrew Gemino
Papers in French
Editor: Michel Kalika

ADMINISTRATIVE PERSONNEL
James P. Tinsley
AIS Executive Director
Vipin Arora
CAIS Managing Editor
University of Nebraska at Omaha
Sheri Hronek
CAIS Publications Editor
Hronek Associates, Inc.
Copyediting by
S4Carlisle Publishing Services