

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

This article appears on pages 273-289 of the Lecture Notes in Computer Science, Vol. 1525, published by Springer-Verlag (1998) and is part of the proceedings for the Second Information Hiding Workshop held in Portland, Oregon, USA, April 15-17, 1998.

Abstract. 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. In this paper, we identify characteristics in current steganography software that direct the steganalyst to the existence of a hidden message and introduce the ground work of a tool for automatically detecting the existence of hidden messages in images.

Introduction

Steganography encompasses methods of transmitting secret messages through innocuous cover carriers in such a manner that the very existence of the embedded messages is undetectable. Creative methods have been devised in the hiding process to reduce the visible detection of the embedded messages. An overview of current steganography software and methods applied to digital images is examined in [JJ98].

Hiding information, where electronic media are used as such carriers, requires alterations of the media properties which may introduce some form of degradation. If applied to images that degradation, at times, may be visible to the human eye [KM92] and point to signatures of the steganographic methods and tools used. These signatures may actually broadcast the existence of the embedded message, thus defeating the purpose of steganography, which is hiding the existence of a message.

Two aspects of attacks on steganography are detection and destruction of the embedded message. Any image can be manipulated with the intent of destroying some hidden information whether an embedded message exists or not. Detecting the existence of a hidden message will save time in the message elimination phase by processing only those images that contain hidden information. Detecting an embedded message also defeats the primary goal of steganography, that of concealing the very existence of a hidden message. Our goal is not to advocate the removal or disabling of valid copyright information from watermarked images, but to point out the vulnerabilities of such approaches, as they are not as robust as is claimed.

In this paper we will look at steganography and watermarking techniques with equal interest. The difference between invisible digital watermarking (or imperceptible to the human eye) and digital steganography is based primarily on intent. Steganography conceals a message where that hidden message is the object of the communication. For example, sending a satellite photograph hidden in another image. Digital watermarks extend some information that may be considered attributes of the cover such as copyright. In the case of digital watermarks, the cover is the object of communication. Sometimes the methods used to embed a watermark and steganography messages are the same. Many digital watermarks are invisible to the human eye, but they may also be known. Watermarking techniques are more robust to attacks such as compression, cropping, and some image processing where the least significant bits are changed. For this reason, digital watermarks are used to embed copyright, ownership, and license information.

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

Invisible watermarking is treated as a subset of steganography though some aspects discussed are unique to tools identified as digital watermarking or steganography tools. When analysis for detection and destruction are applied, the steganography and watermarking tools are treated equally. The intent of this paper is to describe some methods of detecting and destroying hidden messages within computer images. Experimental results are presented to support these claims and identify characteristics in existing steganography software. We will provide a review of various weaknesses in some tools and illustrate how these may be exploited in steganalysis.

The rest of the paper is organized as follows. Section 2 introduces new terminology for steganalysis. Section 3 briefly introduces various methods for embedding information in images and categorizes tools used in this paper. Section 4 introduces some detection methods and identifies unique signatures of steganography tools which reveal the existence of hidden messages. Detection defeats the goal of steganography which is to hide the existence of an embedded message. We will provide examples of some characteristics in a sample of tools and illustrate how these may be vulnerable and exploited in steganalysis. Detection is but one part of steganalysis. Section 5 reveals limitations in the survivability of hidden messages and identifies methods for the destruction of such messages. Destruction methods and examples will be identified. The paper concludes with comments on steganography, steganalysis and related work. A list of additional readings, software, and resources used in researching this topic and additional information on steganography is available at <http://isse.gmu.edu/~njohnson/Steganography>.

Terminology

Steganography literally means "covered writing" and is the art of hiding the very existence of a message. The possible cover carriers are innocent looking carriers (images, audio, video, text, or some other digitally representative code) which will hold the hidden information. A message is the information hidden and may be plaintext, ciphertext, images, or anything that can be embedded into a bit stream. Together the cover carrier and the embedded message create a stego-carrier. Hiding information may require a stegokey which is additional secret information, such as a password, required for embedding the information. For example, when a secret message is hidden within a cover image, the resulting product is a stego-image. A possible formula of the process may be represented as:

$$\text{cover medium} + \text{embedded message} + \text{stegokey} = \text{stego-medium}$$

New terminology with respect to attacks and breaking steganography schemes is similar to cryptographic terminology; however, there are some significant differences. Just as a cryptanalyst applies cryptanalysis in an attempt to decode or crack encrypted messages, the *steganalyst* is one who applies steganalysis in an attempt to detect the existence of hidden information. With cryptography, comparison is made between portions of the plaintext (possibly none) and portions of the ciphertext. In steganography, comparisons may be made between the cover-media, the stego-media, and possible portions of the message. The end result in cryptography is the ciphertext, while the end result in steganography is the stego-media. The message in steganography may or may not be encrypted. If it is encrypted, then if the message is extracted, the cryptanalysis techniques may be applied.

In order to define attack techniques used for steganalysis, corresponding techniques are considered in cryptanalysis. Attacks available to the cryptanalyst are ciphertext-only, known plaintext, chosen plaintext, and chosen ciphertext. In *ciphertext-only* attacks, the cryptanalyst knows the ciphertext to be

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

decoded. The cryptanalyst may have the encoded message and part of the decoded message which together may be used for a *known plaintext* attack. The *chosen plaintext attack* is the most favorable case for the cryptanalyst. In this case, the cryptanalyst has some ciphertext which corresponds to some selected plaintext. If the encryption algorithm and ciphertext are available, the cryptanalyst encrypts plaintext looking for matches in the ciphertext. This *chosen ciphertext attack* is used to deduce the sender's key. The challenge with cryptography is not in detecting that something has been encrypted, but decoding the encrypted message.

Somewhat parallel attacks are available to the steganalyst. These are *stego-only*, *known cover*, *known message*, *chosen stego*, and *chosen message*. A *stego-only attack* is similar to the ciphertext only attack where only the stego-medium is available for analysis. If the "original" cover-media and stego-media are both available, then a *known cover attack* is available. The steganalysis may use a *known message attack* when the hidden message is revealed at some later date, an attacker may attempt to analyze the stego-media for future attacks. Even with the message, this may be very difficult and may even be considered equivalent to the stego-only attack. The *chosen stego attack* is one where the steganography tool (algorithm) and stego-media are known. A *chosen message attack* is one where the steganalyst generates stego-media from some steganography tool or algorithm from a known message. The goal in this attack is to determine corresponding patterns in the stego-media that may point to the use of specific steganography tools or algorithms.

Steganographic Methods

The Internet is a vast channel for the dissemination of information that includes publications and images to convey ideas for mass communication. Images provide excellent carriers for hidden information and many different techniques have been introduced [And96], [BGML96], [JJ98]. A subset of steganography and digital watermarking tools is used in this paper to test detection properties and robustness to manipulations in efforts to destroy or disable the embedded message. These tools can be categorized into two groups: those in the *Image Domain* and those in the *Transform Domain*.

Image Domain tools encompass bit-wise methods that apply least significant bit (LSB) insertion and noise manipulation. These approaches are common to steganography and are characterized as "simple systems" in [AP96]. The tools used in this group include StegoDos [Anon], S-Tools [Bro94], Mandelsteg [Has], EzStego [Mac], Hide and Seek (versions 4.1 through 1.0 for Windows 95) [Mar], Hide4PGP [Rep], Jpeg-Jsteg [Uph], White Noise Storm [Ara94], and Steganos [Hans]. The image formats typically used in such steganography methods are lossless and the data can be directly manipulated and recovered. Including additional components such as masks or image objects to watermark an image is an image domain approach that is somewhat independent of image format.

The *transform domain* grouping of tools include those that involve manipulation of algorithms and image transforms such as discrete cosine transformation (DCT) [CKSL96], [KRZ94] and wavelet transformation [XBA97]. These methods hide messages in more significant areas of the cover and may manipulate image properties such as luminance. Watermarking tools typically fit this categorization and the subset used in this paper is PictureMarc [Dig], JK-PGS [JK], SysCop [Med], and SureSign [Sig]. These techniques are typically far more robust than bit-wise techniques; however a tradeoff exists between the amount of information added to the image and the robustness obtained [JJ98]. Many transform domain methods are independent to image format and may survive conversion between lossless and lossy formats.

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

JPEG images use the Discrete Cosine Transform (DCT) to achieve image compression. The compressed data is stored as integers; however, the calculations for the quantization process require floating point calculations which are rounded. Errors introduced by rounding define the lossy characteristic of the JPEG compression method [BS95]. The tool Jpeg-Jsteg [Uph] is a steganography tool that hides information by manipulating the rounding values of the JPEG DCT coefficients. Information is hidden in the JPEG image by modulating the rounding choices either up or down in the DCT coefficients. Detection of such an embedded message would seem to be quite difficult. (An advantage DCT has over other transforms is the ability to minimize the block-like appearance resulting when the boundaries between the 8x8 sub-images become visible (known as *blocking artifact*) [GW92].)

Some techniques share characteristics of both image and transform domain tools. These may employ patchwork, pattern block encoding [BGML96], spread spectrum methods [CKLS95] and masking [JJ98] which add redundancy to the hidden information. These approaches may help protect against some image processing such as cropping and rotating. The patchwork approach uses a pseudo-random technique to select multiple areas (or patches) of an image for marking. Each patch may contain the watermark, so if one is destroyed or cropped, the others may survive. Masks may fall under the image domain as being an added component or image object. However, a mask may be added to an image by adjusting image properties or transform thus adopting characteristics of transform domain tools.

Detecting Hidden Information

Steganography tools typically hide relatively large blocks of information where watermarking tools place less information in an image, but the watermark is distributed redundantly throughout the entire image [KRZ94]. In any case, these methods insert information and manipulate the images in ways as to remain invisible to the human eye. However, any manipulation to the image introduces some amount of distortion and degradation of some aspect in the "original" image's properties. The tools vary in their approaches for hiding information. Without knowing which tool is used and which, if any, stegokey is used, detecting the hidden information may become quite complex. However, some of the tools produce stego-images with characteristics that act as signatures for the steganography method or tool used.

To begin evaluating images for additional, hidden information, the concept of defining a "normal" or average image was deemed desirable. Defining a normal image is somewhat difficult when considering the possibilities of digital photographs, paintings, drawings, and graphics. Only after evaluating many original images and stego-images as to color composition, luminance, and pixel relationship do anomalies point to characteristics that are not "normal" in other images. Several patterns became visible when evaluating many images used for applying steganography. The *chosen message* and *known cover* attacks were quite useful in detecting these patterns. In images that have color palettes or indexes, colors are typically ordered from the most used colors to the least used colors to reduce table lookup time. The changes between color values may change gradually but rarely, if ever, in one bit shifts. Gray-scale image color indexes do shift in 1-bit increments, but all the RGB values are the same. Applying a similar approach to monochromatic images other than gray-scale, normally two of the RGB values are the same with the third generally being a much stronger saturation of color. Some images such as hand drawings, fractals and clip art may shift greatly in the color values of adjacent pixels. However, having occurrences of single pixels outstanding may point to the existence of hidden information.

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

Added content to some images may be recognizable as exaggerated noise. This is a common characteristic for many bit-wise tools as applied to 8-bit images. Using 8-bit images without manipulating the palette will, in many cases, cause color shifts as the raster pointers are changed from one palette entry to another. If the adjacent palette colors are very similar, there may be little or no noticeable change. However, if adjacent palette entries are dissimilar, then the noise due to the manipulation of the LSBs is obvious [CPL95]. For this reason that many authors of steganography software and some articles stress the use of gray-scale images (those with 256 shades of gray) [Aur95]. Gray-scale images are special occurrences of 8-bit images and are very good covers because the shades gradually change from color entry to color entry in the palette.

Using images with vastly contrasting adjacent palette entries to foil steganography software so that small shifts to the LSBs of the raster data will cause radical color changes in the image that advertise the existence of a hidden message [CPL95]. Without altering the 8-bit palette, changes to the LSBs in the raster data may show dramatic changes in the stego-image:

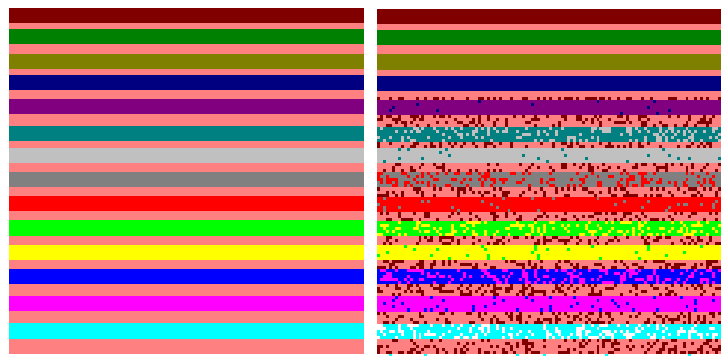


Fig. 1. Original 8-bit cover image (left), and the 8-bit stego-image (right) created with *Hide and Seek*.

Some of the bit-wise tools attempt to reduce this affect by ordering the palette [Mac], [Rep]. Even with a few numbers of distinct colors, sorting the palette is may not be sufficient to keep from broadcasting the existence of an embedded message. Other bit-wise tools and a transform tool take it a step farther and create new palettes [Bro94], [Rep], [Med]. Converting an 8-bit image to 24-bit provides direct access to the color values for manipulation and any alteration will most likely be visually undetectable. The disadvantage is the resulting image is much larger in size and may be unsuitable for electronic transmission. A possible solution is to convert the image back to an 8-bit image after the information is hidden in the LSBs. Even if the colors in the image palette change radically, this method may still hide the fact that a message exists.

Word of caution: since 8-bit images are limited to 256 unique color entries in the image palette, consideration of the number of unique colors used by the image must be considered. For example, if an image contains 200 unique colors and steganography is applied then the number of unique colors could easily jump to 300 (assuming that LSB steganography alters on average 50% of the bits and the new colors are added). Reducing the image to 8-bit again will force the image into 256 colors. There is a high probability that some of the new colors created when modifying the LSBs will be lost.

One method around this is to decrease the number of colors to a value that will maintain good image quality and ensure that the number of colors will not increase beyond 256. This novel approach applies techniques described in [Hec82], [Way96] and reduces the number of colors to no less than

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

32 unique colors. These 32 colors are "expanded" up to eight palette entries by adding adjacent colors in the palette that are very close to the original color [Bro94]. This method produces a stego-image that is so close to the original cover image that virtually no visual differences are detected. However, this approach also creates a unique pattern which will be explored further in this paper.

Looking for signatures

One method for detecting the existence of hidden messages in stego-images is to look for obvious and repetitive patterns which may point to the identification or signature of a steganography tool or hidden message. Distortions or patterns visible to the human eye are the easiest to detect. An approach used to identify such patterns is to compare the original cover-images with the stego-images and note visible differences (known-cover attack). Minute changes are readily noticeable when comparing the cover and stego-images. These subtle distortions may go unnoticed without the benefit of such a comparison. For example, distortion introduced into an image may resemble JPEG compression noise. This "noise" jumps out of the stego-images when compared with the original cover images. Without the benefit of using the cover image, such noise may pass for an integral part of the image and go unnoticed. In making these comparisons with numerous images, patterns begin to emerge as possible signatures to steganography software. Some of these signatures may be exploited automatically to identify the existence of hidden messages and even the tools used in embedding the messages. With this knowledge-base, if the cover images are not available for comparison, the derived known signatures are enough to imply the existence of a message and identify the tool used to embed the message. However, in some cases recurring, predictable patterns are not readily apparent even if distortion between the cover and stego-images is noticeable.

One type of distortion is obvious corruption, as seen in figure1 and discussed in [KM92]. A set of test images was created with contrasting adjacent palette entries as prescribed in [CPL95]. Some of the tools, specifically those in the bit-wise set, produced severely distorted and noisy stego-images [Mac], [Mar], [Hans]. These distortions are severe color shifts that advertise the existence of a hidden message. Detecting this characteristic may be automated by investigating pixel "neighborhoods" and determining if an outstanding pixel is common to the image, follows a pattern, or resembles noise.

Not all of the bit-wise tools produce this type of image distortion. Several bit-wise programs and those in the transform set embedded information without visible distortion of the stego-image. Even though these tools pass this test, other patterns emerged that can be used to detect the possible existence of an embedded message.

8-bit color and gray-scale images which have color palettes or indexes are easier to analyze visually. Tools that provide good results "on paper" may have digital characteristics making the existence of a message detectable [Anon], [Bro94], [Has], [Mar], [Med]. Unlike the obvious distortions mentioned in [KM92] or predicted in [CPL95], some tools maintained remarkable image integrity and displayed almost no distortion when comparing the cover and stego-images on the screen or in print [JJ98], [Joh95]. The detectable patterns are exposed when the palettes of 8-bit and gray-scale images are investigated.

Detecting the existence of a hidden message is accomplished by creating an array of unique pixel values within the image. Then, sort by luminance calculated as follows [BS95]:

$$L = (0.299 \times \text{Red}) + (0.587 \times \text{Green}) + (0.114 \times \text{Blue})$$

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

Investigation of image properties provides promising message detection techniques. Investigating known image characteristics for anomalies point out the possible existence of hidden information. Distortions or patterns visible to the human eye are the easiest to detect especially with the aid of comparing cover images with stego-images. In doing so, a knowledge-base of repetitive, predictable patterns can be established which identifies characteristics that assist in stego-only analysis. Such information assists in automating the detection processes. Such steganalysis tools can identify the existence of hidden messages and even the tools used to embed the messages [San97]. Many bit-wise tools use LSB and similar approaches to hide data. Some times the data is encrypted and other times it is not.

Examples of Palette Signature

S-Tools

In an effort to keep the total number of unique colors less than 256, S-Tools reduces the number of colors of the cover image to a minimum of 32 colors. The new "base" colors are expanded over several palette entries. Sorting the palette by its luminance, blocks of colors appear to be the same, but actually have variances of 1 bit value. The approach is the same with 8-bit color and gray-scale images. When this method is applied to gray-scale cover images, the stego-image is no longer gray-scale as the RGB byte values within a pixel may vary by one bit. This is a good illustration of the limits of the human eye. However, the manner in which the pallet entries vary are uncommon except to a few steganographic techniques.

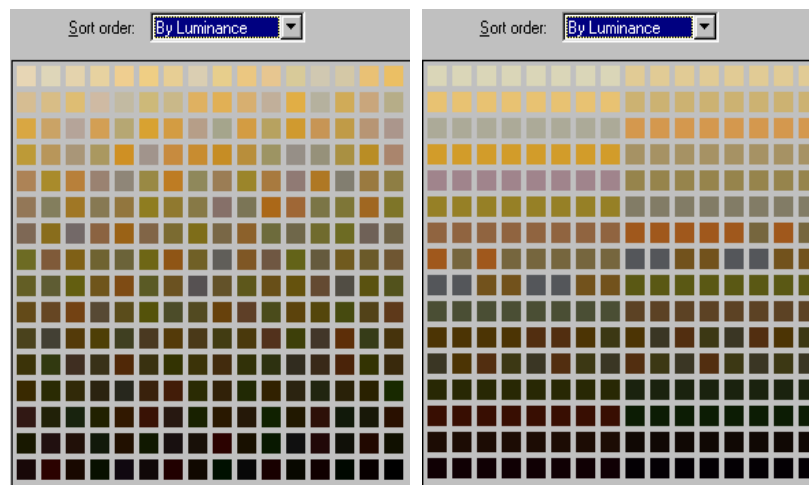


Fig. 2. Cover (left) and stego-image palette (right) after S-Tools

Investigation of many images does not illustrate this pattern occurring "naturally." Gray-scale and other monochromatic images do not follow a similar pattern but the in each step in the palette entry all the RGB values are incremented the same so the pattern does not follow this example. Nor has it been found that images containing large areas of similar colors produce this pattern. Such images contain similar colors but the variance in colors is far greater than that represented by a stego-image produced by S-Tools. Other bit-wise and transform tools share similar characteristics to S-Tools [Has], [Rep], [Med].

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

SysCop

SysCop is the only transform tool that follows this pattern when manipulating 8-bit images. For example, adjacent pallet entries may be 00 00 00, 01 01 00, 01 00 01, etc. If an 8-bit cover image has near 256 colors, SysCop will reduce the number of colors, but not to the degree applied in S-Tools. Detecting a signature through the color variances in the palette for near 256 colors is more difficult than detecting such a pattern in S-Tools. SysCop does, however, typically buffer a number of pallet entries (32+) with black (00 00 00) before the raster data begins. A 256 color approximation of a photograph with black areas rarely has a large number of palette entries with values of (00 00 00). It is far more common for black areas to actually be a number of different colors near black.

Investigating SysCop's manipulation of gray-scale GIF images requires a bit more than casual observation of the palette. In processing gray-scale images, SysCop generates the GIF87a formatted image but with an abbreviated palette table. For example, if the cover image is a GIF87a gray-scale image but only uses nine shades of gray, the image file still has a 256 color index in the file between offset values 0x0D through 0x30C ranging in values from 0 through 255. When SysCop processes the gray-scale files, the palette is reduced to the actual colors used in the stego-image. If nine colors are used in the stego-image, then only nine unique RGB triples are in the stego-image file's palette instead of the expected 256 color index.

Mandelsteg

Viewing the palette also points to the identification of fractal images generated using Mandelsteg. This tool is unique in that it does not manipulate any preexisting cover images, but generates Mandelbrot fractal graphics as cover images for hidden messages. If a file name is passed as a parameter to Mandelsteg, then the file is hidden in the Mandelbrot image. Depending upon the parameters, the image may vary in color and size. All Mandelsteg generated images have 256 palette entries in the color index and all have a detectable pattern in the image palette of 128 unique colors with two palette entries for each color.

Hide and Seek

Hide and Seek creates stego-images with different properties depending upon the version applied. Both versions 4.1 and 5.0 of Hide and Seek share a common characteristic in the palette entries of stego-image. Investigating the palettes of 265 color images, or viewing the histogram, shows that all the palette entries divisible by four for all bit values. This is a very unusual occurrence. Gray-scale images processed by version 4.1 and 5.0 have 256 triples as expected, but the range in sets of four triples from 0 to 252, with incremental steps of 4 (i.e.; 0, 4, 8, ..., 248, 252). A key to detecting this when viewing images casually is that the "whitest" values in an image are 252 252 252. To date, this signature is unique to Hide and Seek. A typical gray-scale color index in a GIF image is from offset 0x0D through 0x30C ranges in triples (the three RGB entries are the same value) from 0 to 255.

In addition to recognizable patterns in the palette, Hide and Seek versions 4.1 and 5.0 have produced characteristics that point to the possible existence of a hidden message. In version 4.1 all files must be 320x480 pixels and contain 256 colors. If the image is smaller than the minimum, then the stego-image is padded with black space. If the cover image is larger, the stego-image is cropped to fit. Version 5.0 allows more sizes to be used for cover images, but the images are forced to fit an exact size. If the image is smaller than the minimum (320x200), then the stego-image is padded with black space. If any image is larger than the nearest allowable image size, then it is padded with black space

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

to the next larger size. If any image exceeds 1024 pixels in width or 768 pixels in height, then an error message is returned. The padded areas are added prior to embedding the message and are thus used in hiding the message. If the padded area is removed from an image, then the message cannot be recovered. Images are forced into sizes 320x200, 320x400, 320x480, 640x400, and 1024x768. StegoDos produces a similar effect as it only works with 256 color, 320x200 images. If images are not this exact size, they are cropped to fit or padded with black space.

In Hide and Seek 1.0 for Windows 95 the image size restrictions are no longer an issue; however, the cover images are still limited to 256 colors or gray-scale. In previous version of Hide and Seek GIF images were used as covers. In the Windows 95 version, BMP images are used due to licensing issues with GIF image compression. No longer do the stego-image palettes produce predictable patterns as in versions 4.1 and 5.0.

Hide4PGP

Hide4PGP uses 8-bit and 24-bit BMP images as cover images and provides a number of options for selecting how the 8-bit palettes are handled or at what bit levels the data is hidden. The default storage area for hidden information is in the LSB of 8-bit images and in the fourth LSB (that is the fourth bit from the right) in 24-bit images. BMP files have a 54 byte header. The raster data in 24-bit images follow this header. Since, 8-bit images require a palette (or color index), the 1024 bytes following header are used for the palette. Hiding plaintext and using the default settings in Hide4PGP, extracting the fourth LSB starting at the 54th byte for 24-bit BMP files and extracting the LSB starting at byte 1078 reveals the hidden plaintext message. If it so happens that the embedded message is encrypted, then cryptanalysis techniques can be applied in attempts to crack the encryption routine.

The options for selecting the bit level to hide information are: 1 for the LSB, 2 for the second LSB, 4 for the fourth LSB, and 8 for the eighth bit. Any of these options produce visible noise in many 8-bit images, so options to manipulate the image palette were added. These greatly improve the look of the resulting cover image but add properties that are unique to steganography and point the viewer to the possibility of a hidden message. Two options allow duplicating palette entries of colors that are more often used and ordering the palette entries to like colors. The number of duplicate entries is always an even number. This is characteristic that can be employed as at signature (similar to [Bro94] and [Med]). By ordering the palette, Hide4PGP, pairs similar colors together similar to the approach of [Mac].

Jsteg-Jpeg

In plotting the coefficients using the IDCT formula of JPEG images, the expected result is a relatively smooth graph for values of not equal to zero. However, plotting the coefficients of images created with Jpeg-Jsteg produce more erratic graphs and show steps resulting from duplicate coefficient values due to exaggerated rounding errors caused by storing the hidden information. This distortion is more noticeable for coefficient values less than zero [Col97].

Destroying Steganography and Watermarks

Detecting the existence of hidden information defeats the goal of imperceptibility. Tools in the transform set are far more difficult to detect without the original image for comparison. Knowledge of an existing watermark may be known so detecting it is not always necessary. A successful attack on a watermark is to destroy or disable it [AP96]. With each of the image and transform domain tools, there

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

is a trade off between the size of the payload (amount of hidden information) that can be embedded and the survivability or robustness of that information to manipulation of the images.

The methods devised by the authors for destruction are not intended to advocate the removal or disabling of valid copyright information from watermarked images, as an illicit behavior, but to evaluate the claims of watermarks and study the robustness of current methods. Some methods of disabling hidden messages may require considerable alterations to the stego-image. Destruction of embedded messages is fairly easy in cases where bit-wise methods are used since these methods employ the LSBs of images which may be changed with compression of small image processes. More effort is required with transform set of data hiding tools since the hidden message is integrated more fully into the cover. A goal for many transform methods is to make the hidden information (the watermark) such an integral part of the image that the only way to remove or disable it is to destroy the stego-image. Doing so will render the image useless to the attacker.

We will illustrate techniques for testing digital watermarks which provide similar functionality to watermarking test tools [Unz], [Kuh97]. Such tools and techniques should be used by those considering making the investment of watermarking to provide a sense of security of copyright and licensing just as password cracking tools are used by system administrators to test the strength of user and system passwords. If the password fails, the administrator should notify the password owner that the password is not secure.

Bit-wise methods are vulnerable to small amounts of image processing. A quick way to destroy many messages hidden with these techniques is to convert the image to a lossy compression format such as JPEG. Recompressing JPEG images processed with Jpeg-Jsteg will destroy the message embedded in the DCT coefficients as they are recalculated. The transform set of techniques that may apply transformations, redundancy, and masking merge the hidden information with integral properties of the images. These methods are more robust than the bit-wise methods, but are still vulnerable to destruction.

Consider this simple formula. Assume a measurement of the threshold of human imperceptibility in an image (t) and the portion above this threshold as image part that contains visible distortion if altered (v). The equation of image (I) is:

$$I = v + t$$

The size of t is available to both the owner watermark and to the attacker. As long as t remains in the imperceptible region, there exists some t' use by the attacker where $I' = v + t'$ and there is no perceptible difference between I and I' . This attack may be used to remove or replace the t region. A variation of this attack is explored in [CMYY97] to the aspect of counterfeiting watermarks. If information is added to some media such that the added information cannot be detected, then there exists some amount of additional information that may be added or removed within the same threshold which will overwrite or remove the embedded covert information.

A series of image processing tests were devised to evaluate the robustness threshold of the bit-wise and transform tools. These tests will eventually alter the hidden information to the point that it cannot be retrieved. This fact may be viewed as a weakness of the "reader" instead of the "writer" in some of these tools. The motivation behind these tests is to illustrate what the techniques will withstand and what are some common vulnerabilities. The method of testing and measuring each tool consisted of using existing images and creating new images for testing. The images include digital photographs,

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

clip art, and digital art. The digital photographs are typically 24-bit with thousands of colors or 8-bit grayscale. JPEG and 24-bit BMP files make up the majority of the digital photographs. Clip art images have relatively few colors and are typically 8-bit GIF images in our experiment. Digital art images are not photographs, but may have thousands of colors. These images may be 24-bit (BMP or JPEG) or 8-bit images (BMP or GIF). Where necessary, images were converted to other formats as specified by the steganography or watermarking tool requirements.

A number of images from each type were embedded with known messages or watermarks and the resulting stego-images were verified for the message contents. In the robustness testing, the stego-images are manipulated with a number of image processing techniques and checked for the message contents. The tests include: converting between lossless and lossy formats, converting between bit densities (24-bit, 8-bit, grayscale), blurring, smoothing, adding noise, removing noise, sharpening, edge enhancement, masking, rotating, scaling, resampling, warping, converting from digital to analog and back (printing and scanning), mirroring, flipping, adding bit-wise messages, adding transform messages, and applying the unZign and StirMark tools to test the robustness of watermarking software. A series of tests were also performed to determine the smallest images that can be used successfully to hide data for each tool.

Minor image processing or conversion to JPEG compressed images was sufficient to disable the bit-wise tools. The transform methods survived a few of the image processing tests. Many images were used for each test as results varied between the use of 8-bit, 24-bit, lossless and lossy image formats. All tests were conducted using *Paint Shop Pro* by JASC and the results were recorded on whether the hidden information was detected and recovered with each steganography and watermarking tool. PictureMarc was added to images via Adobe *Photoshop*[®]. SureSign was added to images using both *Paint Shop Pro* and *Photoshop*.

With any one of the tests, tools that rely on bit-wise methods to hide data failed to recover any messages. The transform tools such as survived many of these tests, but failed with combinations of these image processes. Existing tools were also applied to the stego-images to test robustness [Unz], [Kuh97]. The observed success in making the watermark unreadable is in introducing small geometric distortions to the image then resampling and smoothing. This combines the effects of slight blurring, edge enhancement, and asymmetric resizing (warping). These combinations are very effective in reducing the ability for watermarking tools to identify the embedded watermark. Companies such as Digimarc and Signum Technologies maintain that even with severe image manipulation, the watermark may be recovered if both the altered watermarked image and the original image can be used together to extract the partially destroyed watermark.

An attractive feature in the use of watermarking technology in the Internet is the ability to use a software robot (softbot or spider) that searches through web pages for watermarked images. If watermarks are found, the information can be used to identify copyright infringement [Dig]. An attack that illustrates the limitation of such a softbot takes advantage of the image size limitations of a readable watermark by splitting the watermarked image into sufficiently small pieces so the watermark reader cannot detect the watermark [PAK98]. This method does not attack the processing of an image to embed or remove a mark, but illustrates a way to bypass detection.

Related Work

This paper provided an introduction to steganalysis and identified weaknesses and visible signs of steganography. This work is but a fraction of the steganalysis approach. To date a general detection

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

technique as applied to digital image steganography has not been devised and methods beyond visual analysis are being explored. Too many images exist to be reviewed manually for hidden messages. We have introduced some weaknesses of steganographic software that point to the possible existence of hidden messages. Detection of these "signatures" can be automated into tools for detecting steganography [San97]. *Stegodetect* takes advantage of palette patterns and the characteristics, and analyzes pixel "neighborhoods." Tools for detecting hidden information are promising for future work in steganalysis and for verifying watermarks.

Steganography pertains not only to digital images but also to other media, including voice, text and binary files, and communication channels. The ease in use and abundant availability of steganography tools has law enforcement concerned in trafficking of illicit material via web page images, audio, and other files being transmitted through the Internet. Methods of message detection and understanding the thresholds of current technology are necessary to uncover such activities. Ongoing work in the area of Internet steganography [Dun] investigates embedding, recovering, and detecting information in TCP/IP packet headers and other network transmissions.

Development in the area of covert communications and steganography will continue. Research in building more robust digital watermarks that can survive image manipulation and attacks continues to grow. The more information is placed in the public's reach on the Internet, the more owners of such information need to protect themselves from theft and false representation. Success in steganographic secrecy results from selecting the proper mechanisms. However, a stego-image which seems innocent enough may, upon further investigation, actually broadcast the existence of embedded information.

Comments and Conclusion

Steganography transmits secrets through apparently innocuous covers in an effort to conceal the existence of a secret. Digital image steganography and its derivatives are growing in use and application. In areas where cryptography and strong encryption are being outlawed, citizens are looking at steganography to circumvent such policies and pass messages covertly. Commercial applications of steganography in the form of digital watermarks and digital fingerprinting are currently being used to track the copyright and ownership of electronic media. Understanding and investigating the limitations of these applications helps to direct researchers to better, more robust solutions. Efforts in devising more robust watermarks are essential to ensure the survivability of embedded information such as copyright and licensing information. Tools that test the survivability of watermarks are essential for the evolution of stronger watermarking techniques. Using these tools and methods described in this paper, potential consumers of digital watermarking tools, can see how much (or how little) effort is required to make the watermark unreadable by the watermarking tools.

Perhaps an inherent weakness in many watermark approaches, is the advertisement that an invisible watermark exists in a file. With steganography, if the embedded message is not advertised, casual users will not know it even exists and therefore will not attempt to remove the mark. However, advertising the fact that some hidden information exists, is only an invitation to "crackers" as a challenge. Some watermarking tools are distributed with over-the-shelf software, such as Adobe Photoshop® [Dig]. A method was recently advertised over the Internet that such a tool has been "cracked" and showed how to watermark any image with the ID of someone else. Almost any information can be added which can even be used to overwrite valid watermarks with "forged" ones. If humanly imperceptible information is embedded within a cover, then humanly imperceptible alterations can be made to the cover which destroys the embedded information.

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

Acknowledgement

The authors would like to thank Eric Cole, Prof. Zoran Duric and others to their contribution in reviewing and commenting on this paper, and David Sanders for his preliminary work in developing a steganalysis tool.

References

- [And96] Anderson, R., (ed.): Information hiding: first international workshop, Cambridge, UK. Lecture Notes in Computer Science, Vol. 1174. Springer-Verlag, Berlin Heidelberg New York (1996)
- [AP96] Anderson, R., Petitcolas, F.: On the Limits of Steganography, IEEE Journal on Selected Areas in Communications, Vol. 16, No. 4, May (1998) 474-481.
- [Aur95] Aura, T.: Invisible Communication, EET 1995. Technical Report, Helsinki Univ. of Technology, Finland, November 1995, http://deadlock.hut.fi/ste/ste_html.html (1995)
- [BGML96] Bender, W., Gruhl, D., Morimoto, N., Lu, A.: Techniques for Data Hiding. IBM Systems Journal Vol. 35, No. 3&4. MIT Media Lab (1996) 313-336.
- [BS95] Brown, W., Shepherd, B.J.: Graphics File Formats: Reference and Guide. Manning Publications, Greenwich, CT (1995)
- [CKLS95] Cox, I., Kilian, J., Leighton, T., Shamoon, T.: Secure Spread Spectrum Watermarking for Multimedia. Technical Report 95-10, NEC Research Institute (1995)
- [CKSL96] Cox, I., Kilian, J., Shamoon, T., Leighton, T.: A Secure, Robust Watermark for Multimedia. In: [And96] (1996) 185-206
- [Col97] Cole, E.: Steganography. Information System Security paper, George Mason University. (1997)
- [CMYY97] Craver, S., Memon, N., Yeo, B., Yeung, N.M.: Resolving Rightful Ownerships with Invisible Watermarking Techniques. Research Report RC 20755 (91985), Computer Science/Mathematics, IBM Research Division (1997)
- [CPL95] Cha, S.D., Park, G.H., Lee, H.K.: A Solution to the Image Downgrading Problem. ACSAC (1995) 108-112
- [Dun] Dunigan, T.: Work in progress on Internet steganography which involves hiding, recovering, and detecting info hidden in the TCP/IP packet headers. **Oak Ridge National Laboratory**, Oak Ridge, TN.
- [GW92] Gonzalez, R.C., Woods, R.E.: Digital Image Processing. Addison-Wesley. Reading, MA, (1992)
- [Hec82] Heckbert, P.: Color Image Quantization for Frame Buffer Display. ACM Computer Graphics, vol. 16, no. 3. July (1982) 297-307.
- [JJ98] Johnson, N.F., Jajodia, S.: Exploring Steganography: Seeing the Unseen. IEEE Computer. February (1998) 26-34
- [Joh95] Johnson, N.F.: Steganography. Information System Security paper, George Mason University (1995) <http://isse.gmu.edu/~njohnson/stegdoc/>

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

[KRZ94] Koch, E., Rindfrey, J., Zhao, J.: Copyright Protection for Multimedia Data. Proceedings of the International Conference on Digital Media and Electronic Publishing, December 1994. Leeds, UK (1994)

[KM92] Kurak, C., McHugh, J.: A Cautionary Note On Image Downgrading. IEEE Eighth Annual Computer Security Applications Conference (1992) 153-159.

[PAK98] Petitcolas, F., Anderson, R., Kuhn, M.: Attacks on Copyright Marking Systems. Second Workshop on Information Hiding, Portland, Oregon, April. This proceedings (1998)

[Pfi96] Pfitzman, B.: Information Hiding Terminology. In: [And96] 347-350

[Way96] Wayner, P.: Disappearing Cryptography. AP Professional, Chestnut Hill, MA (1996)

[XBA97] Xia, X, Boncelet, C.G., Arce, G.R.: A Multiresolution Watermark for Digital Images. IEEE International Conference on Image Processing, October 1997 (1997)

Steganography Software References

Many other software applications are available that provide steganographic results. The following list gives a sample of software available for the PC platform. Every effort is being made to credit the authors of the software reviewed in this paper. However, some authors wish to remain anonymous. Additional software sources are listed at <http://isse.gmu.edu/~njohnson/Steganography>.

Image Domain Tools

[Anon] Anonymous, Author alias: Black Wolf.: *StegoDos - Black Wolf's Picture Encoder v0.90B*, Public Domain. <ftp://ftp.csua.berkeley.edu/pub/cypherpunks/steganography/stegodos.zip>.

[Ara94] Arachelian, R.: *White Noise Storm™ (WNS)*, Shareware (1994)
<ftp://ftp.csua.berkeley.edu/pub/cypherpunks/steganography/wns210.zip>.

[Bro94] Brown, A.: *S-Tools for Windows*, Shareware 1994.
<ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/s-tools3.zip> (version 3),
<ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/s-tools4.zip> (version 4)

[Has] Hastur, H: *Mandelsteg*, <ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/>

[Mac] Machado, R.: *EzStego, Stego Online, Stego*, <http://www.stego.com>

[Mar] Maroney, C.: *Hide and Seek*, Shareware.
<ftp://ftp.csua.berkeley.edu/pub/cypherpunks/steganography/hdsk41b.zip> (version 4.1),
<http://www.rugeley.demon.co.uk/security/hdsk50.zip> (version 5.0), <http://www.cypher.net/products/>
(version 1.0 for Windows 95)

[Rep] Repp, H.: *Hide4PGP*, <http://www.rugeley.demon.co.uk/security/hide4pgp.zip>

[Hans] Hansmann F.: *Steganos*. Deus Ex Machina Communications. <http://www.steganography.com>.

Steganalysis of Images Created Using Current Steganography Software

Neil F. Johnson and Sushil Jajodia

Transform Domain Tools

[Dig] Digimarc Corporation: *PictureMarc™*, *MarcSpider™*, <http://www.digimarc.com>

[JK] Kutter, M., Jordan, F.: *JK-PGS (Pretty Good Signature)*. Signal Processing Laboratory at Swiss Federal Institute of Technology (EPFL). http://ltswww.epfl.ch/~kutter/watermarking/JK_PGS.html

[Med] MediaSec Technologies LLC.: *SysCop™*, <http://www.mediasec.com/>

[Sig] Signum Technologies, *SureSign*, <http://www.signumtech.com>

[Uph] Upham, D.: *Jpeg-Jsteg*. Modification of the Independent JPEG Group's JPEG software (release 4) for 1-bit steganography in JFIF output files. <ftp://ftp.funet.fi/pub/crypt/steganography>.

Watermark and Steganography Analysis and Testing Tools

[Kuh97] Kuhn, M.: *StirMark*.
http://www.cl.cam.ac.uk/~fapp2/watermarking/image_watermarking/stirmark (1997)

[San97] Sanders, D.: *Stegodetect*. Steganography detection tool (1997)

[Unz] Anonymous: *unZign*. Watermarking testing tool available at <http://altern.org/watermark/> - the author may be contracted through unzign@hotmail.com (1997)