Hiding in Plain Sight:

Scrubbing Secret Content from Files

Submission Type: Full Paper

Charles Wood
Duquesne University
woodc@duq.edu

Abstract

Steganography is a technique used to hide encrypted messages within multimedia files. This technique was recently reported to have been used by Osama Bin Laden to communicate with terrorist cells within the United States, and, thus, prevention of the transmission of steganographic content is of great interest to those interested in information security. Methods of steganalysis have been developed that attempt to detect files that contain steganographic content. However, authors of these methods admit that they are not viable for production or have been shown to be defeated by newer advances in steganography. This article illustrates an innovation in which algorithms neutralize any hidden messages without significantly detracting from the underlying integrity of the multimedia files and without the need for prior detection of steganographic content.

Keywords (Required)

Stgeganography, Encryption, Information Security

Introduction

Steganography is the science of embedding messages within other messages. The sole purpose of steganography is to transmit messages between two parties with limited or no possibility of detection from a third party. The term "Steganography" (literally translated from the Greek phrase meaning "covered writing") was first used in the fifteenth century, where messages were sent on the back of wax writing tables or even on the stomachs of rabbits (Kessler 2004). Newer software packages, such as SysCop, Steghide, Mandelsteg, Hide and Seek, Hide4PGP and OpenPuff, to name a few, easily allow users to embed secret messages within multimedia files. The techniques used by these tools can widely vary.

Ker, et al. (2013) describe the steganographic process where an individual (known as the steganographer) wishes to send a covert communication (known as the payload) to a recipient. The steganographer has access to unaltered images (known as covers). The communication channels used by the steganographer (e.g., email, Internet postings, etc.) are monitored by a person who tries to prevent secret messages from being sent (known as the steganalyst). The goals of the steganographer are threefold: (a) to encrypt a secret message and send it to some recipient, (b) to not only hide such information from the steganalyst, but to even hide the fact that a message has been sent, and (c) to resist future audit attempts that would detect the secret message.

A problem exists in that steganography can be used for terrorist, espionage, or criminal activities. Steganography packages are easy to use, and criminal or terrorist organizations have been reported to use steganography in order to secretly transfer information that is hidden from law enforcement and intelligence agencies. For example, multiple sources from the multiple governments indicate that Osama Bin Laden used steganography to send messages to terrorist cells within online multimedia files (McCullough 2001), and a Italian newspaper, Corriere della Sera, cites how court documents and investigative interviews detailing how an al-Qaeda cell which had been captured at the Via Quaranta mosque in Milan used pornographic images on their computers, and that these images had been used to hide secret messages through steganography.

The potential of the use of steganography can also be used to criminally transfer secret information, such as trade secrets, private identity and credit card information, and secret corporate strategies. Secret messages can include matters of personal identity theft, national security and etc., and can pass messages undetected from computer to computer unless there is a tool to catch and remove (or neutralize) the hidden message. The damages that can occur range from financial loss - possibly in the extreme - to the ultimate loss of life, possibility of war, and the possibility of other horrific tragic outcomes. (Kessler 2004; Johnson and Jajodia 1998).

Current wisdom in the fields of information systems and of computer science embrace the acknowledgement of the threat that steganography poses businesses and national security, and also recognizes the importance of detecting the presence of steganographic alteration when and where it appears (termed "steganalysis" by researchers and practitioners). However, detection methods are still in their infancy, and steganographic methods are now designed to bypass existing steganalysis techniques (Open Puff, 2015).

This research is design science (Hevner, et al. 2004) which proposes the building of a construct to solve an existing research problem. This research uses a methodology used in other design science research where existing alternatives compared to a new innovation, where the new innovation addresses some of the weaknesses of existing alternatives (March and Story 2008; Lee, et al. 2008). Specifically in this case, a prototype is developed that purports to solve the problems generated by steganography, after describing the weaknesses inherent in current steganalysis alternatives. This research illustrates a proof of concept that capitalizes on the limitations on visual acuity within human beings to remove hidden steganographic content from within files *without prior detection* (required by steganalysis) of the hidden content while leaving the images virtually unchanged to the human eye.

A Basic Discussion of the Steganographic Method

Steganography can be accomplished by the manipulation of the *least significant bit* (LSB) within some multimedia files, such as BMP or GIF image files, or AU or WAV sound files (Johnson and Jajodia 1998). The concept is that the color (red, green, and blue components, gray-scale components, etc.) or the sound (volume control, pitch, etc.) can be slightly modulated so that the difference is not detectable to the human eye or ear. Within this modulation, a secret message can be placed so as to be undetectable.

To illustrate, observe the two gray colors ("Old Color" and "New Color") in Figure 1.

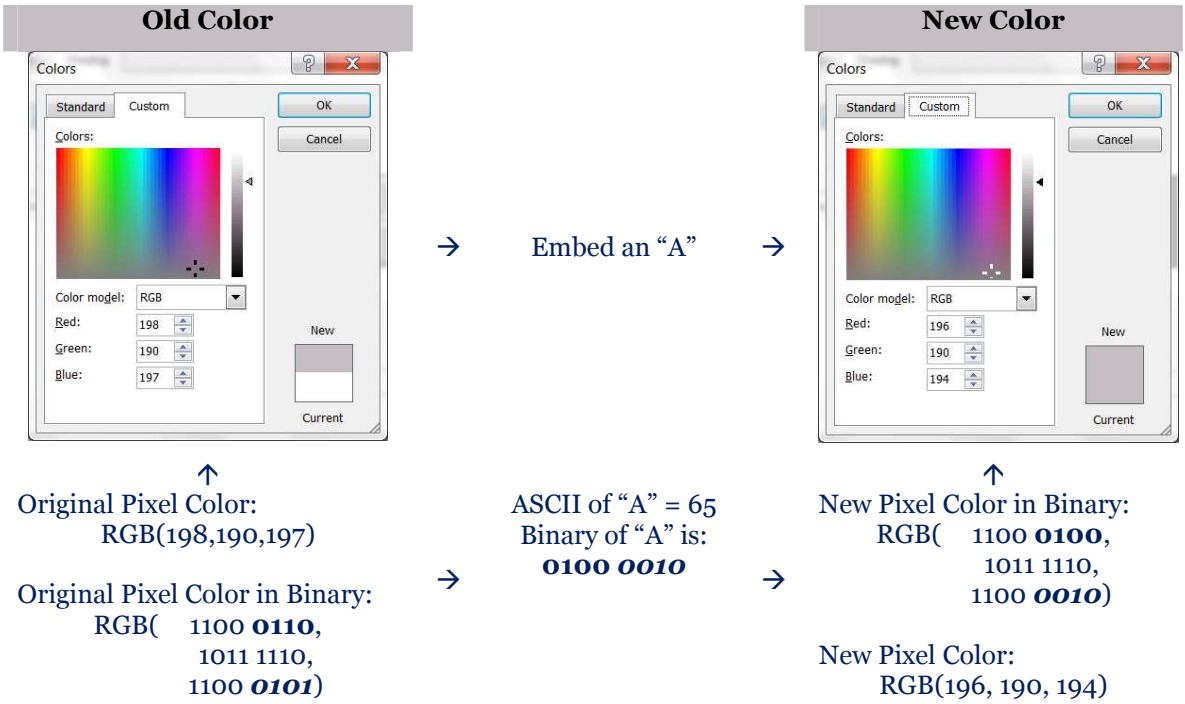

Figure 1 – Embedding an "A" inside a pixel

In Figure 1, a single pixel (showing a gray color) is selected and embedded with the letter "A". Each pixel is made up of three light components – a red component, a green component, and a blue component. The original gray color has red-green-blue (RGB) values, respectively, of 198, 190, and 197. Computers "think" in binary, and the binary representation of the RGB values is 1100 0110, 1011 1110, 1100 0101. As one can see in Figure 1, the difference between the "Current" color and the "New" color is undetectable by the human eye.

Letters are converted to numbers within any computer system. With many, ASCII coding is used. With this coding, the letter "A" is assigned the value of 65 in decimal, or 0100 0010 in binary.

To embed the letter "A" in the pixel, the last four bits of the red and blue components are replaced with the letter "A"'s components. (In Figure 1, the bold and italics parts of "A" replace the corresponding bold and italics parts of the gray pixel, leading to the new color that has RGB components of 196, 190, and 194. These new numbers are very close to the old numbers, and it is unlikely that the casual observer will be able to tell the difference between the two numbers. This is because the most significant bits control most of the color, and the least significant bits are used to fine-tune the color. Because the fine-tuning of the new color is so slight, the human eye cannot often distinguish between such slight color changes in a picture, which is often made up of millions of pixels. Note, too, that we added an "A" to a single pixel by manipulating the least significant four bits per pixel (bpp), but there's no reason to force one pixel per character in order to only manipulate the last LSB. In fact, many steganography tools do indeed embed letters in a secret message over several pixels (<1 bpp) to reduce damage to the multimedia file and also reduce the chance of visual detection. (Although Kawaguchi and Eason (1998) propose a BPCS (Bit-Plane Complexity Segmentation) steganographic method that could embed a secret message in noisy parts of an image using up 4 bpp without observable visual impairment of the image.)

Steganography is also very difficult to detect. Detection methods that can be used for the simple example shown in Figure 1 have to determine that an "A" is embedded in the new pixel, rather than determining that RGB(196, 190, 194) is the color it is supposed to be in the image. Add on top of this that secret messages are often encrypted and compressed before embedding them to the multimedia file, and detection becomes very difficult indeed. This work proposes a programmatic structure to eliminate any files that have steganographic messages embedded within identifiable file types, even if identification of the message or even detection that a message exists is impossible, without harming the content of any multimedia (or other) file in any way that is detectable to the end user.

Advanced steganographic algorithms allow the embedding of a secret message within a larger multimedia file or files, such as pictures, audio files, and video files. When a message is embedded, there is *some* alteration of the original message at the pixel level, but each pixel is altered so slightly so that a secret message can be contained within the picture. (This image with a secret message embedded in it is often called a "stego-image".) Most often, the alteration is so slight that the human eye cannot detect the changes that result from the hidden message.

Steganography can exist within many different types of files, by passing secret messages within, images (by manipulating the values of the RGB field or the luminance (brightness) inside a JPEG image file), audio/sound files (by manipulating the volume control), and even Microsoft Word documents or Adobe Acrobat files (by manipulating the meta data, such as internal file comments, line spacing, letter size or spacing of punctuation, etc.). Even programs can be suspect, by implementing a number of NOPs ("No Operation") within a file that correspond to a message, or by placing a secret message within an executable and adding code to branch around the secret message, thereby making the executable code work exactly as before, but containing a secret message. Other techniques involve adding a secret message before a beginning of file (BOF) marker or after an end of file (EOF) marker, so that users of the file or program will not be able to detect the secret message.

The Failure of Steganalysis

There have been many authors that have examined the implementation of steganography and have discussed the implications to security. For example, Johnson and Jajodia (1998) describe steganographic tools that existed at that time and how such tools pose a threat to law enforcement (by allowing the undetectable trafficking of illicit material) and to company secrets (by allowing the covert transmission of trade secrets) and even national security (by allowing the covert transmission of top secret information).

Steganalysis is the term used to describe the process of detecting secret messages made from steganography. Steganalysis techniques used to detect the existence of steganography have also been examined. Although it may be counter-intuitive, a steganographic technique is known to be "stego-secure" against a steganalysis technique (or against all existing steganalysis detection from any technique) if the steganalysis cannot detect that an image contains a secret message. A steganographic system is considered stego-secure if no statistical test can distinguish between unaltered files and files containing secret messages.

There methods of detection often discussed in the literature are χ^2 steganalysis, Regular/Singular (RS) steganalysis, and most recently, machine learning systems. This section will discuss the weaknesses of each method and how each staganalytic method was circumvented after being developed.

χ^2 Steganalysis

Westfeld and Pfitzman (1999) developed a detection technique that works when pixels are sequentially changed to embed a secret message, commonly referred to as the χ^2 (chi-square) steganalysis. They take advantage of the fact that a pixel stream can be regarded as a high-order Markov process (Philip and Joseph 2000), where the binary value of each pixel is independent from the other pixels. Thus, entropy is maximized when different pixel values have equal probability of occurring.

χ^2 steganalysis starts with a palette image, and counts the colors for each possible pixel, where you have, at most, 256 colors (C={c_0, c_1,...,c_{255}}) contained in each palette. χ^2 steganalysis then uses pairs of values (PoVs) of palette values, so 256 possible colors make 128 pairs of adjacent pixels (k={0-1, 2-3, 4-5,...,254-255}). Next, a count of each occurrence of every possible color is made. (For color c_i, n_i is count of the pixels that contain that color.) The following equation defines the expected frequency of pixel colors as the average pixel count between two adjacent pixels:

$$n'_i = \frac{n_i + n_{i+1}}{2}$$

The χ^2 statistic with k-1 degrees of freedom developed by them to test if equality of n'_i and n_i are equal. As messages are embedded, n'_i and n_i tend to diverge:

$$\chi^2_{k-1} = \sum_{j=1}^{k} \frac{\left(n_{2j} - n'_{2j}\right)^2}{n'_j}$$

The χ^2 test is a strong test to detect embedded content (Westfeld and Pfitzman 1999; Provos and Honeyman 2003). However, steganographers quickly discovered that a way to circumvent detection was to add a secret message to an image file without using sequential pixels, but rather using a pseudo-random order of pixels usually determined by the key chosen by the steganographer (Provos 2001).

The circumvention of χ^2 –based steganalysis is exacerbated by many security tools that claim to detect steganographic content, but use only the χ^2 –test as their technique. Many extant steganography tools boast that they can beat this test (e.g. OpenPuff 2015). Unsuspecting information security officers may not have the training to understand that they are still vulnerable.

Regular/Singular (RS) Steganalysis

RS steganalysis is proposed by Fridrich et al. (2001) as a method of detecting the existence of steganography by examining the flipping of the LSB. Fridrich et al. define three types of flipping: positive flipping (where an LSB is shifted from zero to one during steganalysis), negative flipping (where an LSB is shifted from one to zero during steganalysis) and zero flipping (where an LSB is unchanged during steganalysis). The relationship between positive, negative, and zero flipping is formally defined as follows:

$$F_{-1} = F_1 (F_0 + 1) - 1$$

where F_{-1}, F_1, and F_0, are denoted, respectively, as negative, positive, and zero flipping functions.

RS steganalysis is accomplished using a two-step process. In the first step, the image is divided into separate blocks of pixels, each arranged in a vector G_B in zigzag scan order:

$$G_B = (x_1, x_2, \ldots x_n,)$$

The spatial correlations of LSBs of adjacent pixels can be determined by the discrimination function:

$$f\left(G_B\right) = \sum_{i=1}^{n-1} \left| x_i - x_{i+1} \right|$$

where x_i is the pixel value and n is the number of pixels. The value returned by the discriminant function shown above is inversely proportional to the strength of the correlation (i.e. a small f is indicative of a strong correlation).

In the second step, non-negative flipping and non-positive flipping are applied on each block:

$$M_+(1), M_+(2), \ldots M_+(n) \in \{0,1\}$$
$$M_-(1), M_-(2), \ldots M_-(n) \in \{0,-1\}$$

The relative number of "Regular" positive blocks is denoted as R_m while the relative number of "Singular" positive blocks is denoted as R_m. Conversely, the relative number of "Regular" negative blocks is denoted as R_{-m} while the relative number of "Singular" negative blocks is denoted as. S_{-m}. Fridrich, et al. (2001) show that the R and S blocks have the relationships :

$$R_m \cong R_{-m} \quad \text{and} \quad S_m \cong S_{-m}$$

Fridrich, et al. show that the difference between R_m and R_{-m} and between S_m and S_{-m} both increase with the length of the embedded message. Further, they show that RS steganalysis is resistant to steganographic methods that defeat χ^2 steganalysis.

However, relatively soon after RS steganalysis was introduced, steganographic techniques that utilize a pseudo-chaotic adjustment of LSBs followed by a compensation of the stego-image that ensures the relationship of $R_m \cong R_{-m}$ and $S_m \cong S_{-m}$ remain true (Luo et al. 2007).

Steganalysis using Blind Calibration and Machine-Learning Methods

Thus far, no universal models of steganalysis, where the staganalytic method works across all steganographic embedding algorithms, exist. However, in response to acceptable universal models, recent steganalysis algorithms have been successful through blind self-calibration method (Solanki et al. 2007; Pevný and Fridrich 2010; Dabeer et al. 2004). Recently, such self-calibration are considered to be the best techniques available, in that the steganalyst can determine one out of six steganographic algorithms employed for hiding with a detection accuracy of more than 95% in most cases, even at low embedding rates (> .05 BPNZ) (Solanki et al. 2007).

However, newer steganographic techniques have taken on blind steganalysis. Probably the most promising is the work done by Filler, Judas, and Fridrich (2011) that employs an algorithm based on Syndrome-Trellis Codes that can deliver undetectable messages within .2 BPP. This means that, within an 800x600 gray scale pixel image, the steganographer can deliver a message that 96,000 bytes long, or embed a file of that size. JPEG maps and instructions, client lists, blueprints in .GIF format, etc., can often easily fit within these parameters. Furthermore, with color files or movie files, the maximum undetectable size of a secret file can increase dramatically.

The Problem with Steganographic Procedures that Defeat Steganalysis

There has been much research devoted to defeating steganalysis with the goal of developing stego-secure methods that defeat existing steganalysis (e.g., Wang et al. 2010, Luo et al. 2007). Li (2011) describes multiple steganographic techniques and the steganalysis that was relatively quickly developed to defeat them, and then the response to the steganalysis through the development of more sophisticated steganography.

Top researchers in the field agree that staganalytic methods do not exist that work with unknown algorithms, multiple objects, and multiple actors, and existing steganography research often concentrates on gray-scale JPEG images, to the exclusion of many other file types and multimedia files (e.g., Ker et al., 2013). Indeed, most researchers who have developed staganalytic programs would agree that their detectors work well only in laboratory conditions, and if such detectors were implemented in a business environment, their accuracy would be uneven and unreliable even in situations where older steganographic techniques that have shown to be detectable are used (Ker et al., 2013).

The take away from this section is that modern techniques for detecting secret messages embedded in multimedia files through steganography have yet to be addressed by researchers, and not only are beaten in relatively short order by newer, better steganographic algorithms, but also that the steganalysis techniques often concentrate on solely one type of existing steganography algorithms and also are not robust enough to implement within a production environment (Ker et al., 2013). When one considers that steganography attacks can involve only one or two images posted online or emailed and can pose a threat to national security or a catastrophic event to businesses by allowing the undetectable and illicit transfer of information that can bypass law enforcement or national security defenses. As such, there is need for an innovation that removes the threat of steganography, *yet does not rely on steganalysis* to first detect the steganography.

The Proposed Innovation

According to current wisdom prior to the present invention, any steganographic images or other data file alterations are neutralized or rendered benign only after the steganographic data file alteration has first been identified. Information system security through steganalysis is primarily understood in research to include the sequential steps of discovering (first) and then preventing (second) steganographically inserted messages in data files, with the believed-to-be key initial step of discovery or detection of the presence of the steganographically generated message.

The detection of a steganographically embedded secret message is the necessary first step in any proposed intervention to halt transmission of a message and to remove that secret message from an applicable data file. In this instance, tools that detect steganographically-embedded secret messages prior to decoding or neutralizing them are enormously cumbersome and inefficient for the preponderance of computer security applications for which neutralization of steganography is needed. Equally importantly, too, prior steganalysis approaches have not been designed to address virtually all or all data file types to create a ubiquitous tool for computer security.

However, it is the contention of this research the vast majority of computer files that are transferred over the Web and that need to be free of steganographic alteration also do not need their hidden messages to be detected and decoded simply need to be rendered safe from alteration. This research proposes a *scrubbing tool* that can be applied ubiquitously to a single data files or program or across all data files and programs to remove any secret message, if present, with the advantage in efficiency and effectiveness of deliberately not investigating whether a secret message is present first through these key characteristics:

- The proposed scrubbing tool does not rely on steganalytical methods to detect the presence of steganographically-embedded secret messages.
- The proposed scrubbing tool automatically scrubs, or neutralizes, steganographically-embedded secret messages in data files or across data files, if such secret messages are present.
- The proposed scrubbing tool acts without observable changes to the viewing of the file(s) if the file(s) do not contain a secret message.

Figure 2 shows this graphically. What is needed is a way to (a) keep bad content from reaching our secure areas, (b) keep secret content from leaving our secure areas, and (c) to accomplish this without the need for prior detection.

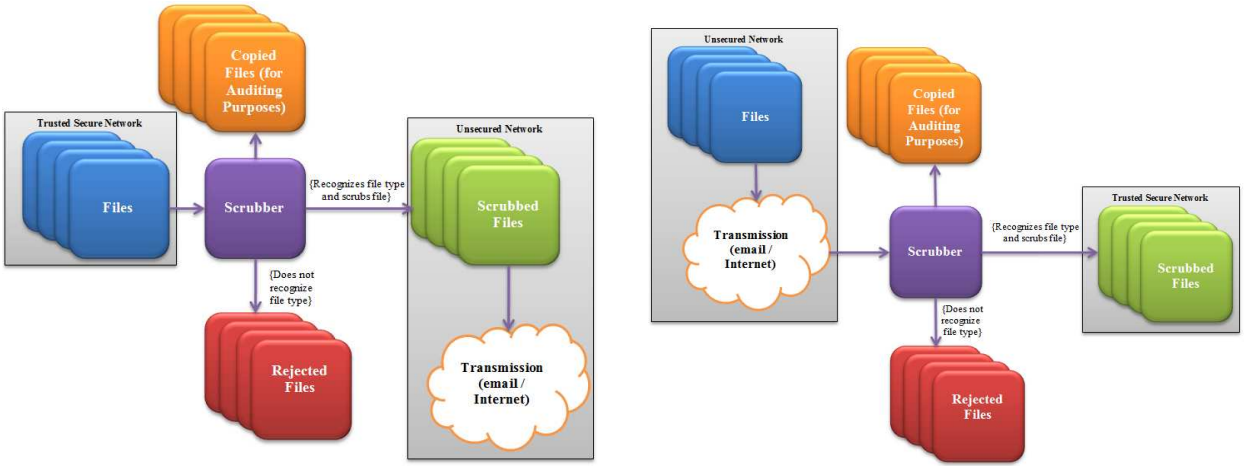

Figure 2 – Files need to be scrubbed coming in or going out of a system

Removal of secret content without prior steganalytic detection of (a) what that secret content is and (b) what method was used to place it in the file, is key to a resolution approaching the system described in Figure 2. At first may seem to be an insurmountable problem, but in truth, it may be accomplished in the same way that secret steganographic messages are first added to a file. For instance, just as a steganographer may add secret content by adjusting the least significant bit in the picture file so that the altered file is indistinguishable from the original, we can erase any content by scrambling the least significant bit. Mathematically, this can be expressed as:

$$bit_{ij} = bit_{ij} \; xor \; RAND(0,1)$$

In the above equation, an exclusive or bit operation (and AND or an OR bit operation can also be used) is applied to a random binary number and the least significant bit in each pixel in a picture of *i* x *j* pixels. The result would be that (a) any secret message that employed these least significant bits would be destroyed, (b) the picture would be altered so slightly so that any changes to the picture would be imperceptible to the human eye, and finally (c) any distortion of the picture done by a scrubbing routine that employs this would not be repeated by multiple applications of the same routine, so that a file exposed to multiple applications of the same routine would not be damaged further after the first iteration.

This code is most easily expressed on a bitmap (BMP) file. After a file is read in, the code to an RGB image file, the following Java code can be used to scrub the image:

```
// Scrubs an image that is loaded into RGB memory
    public final BufferedImage scrub(BufferedImage image) {
        for (int x = 0; x < image.getWidth(); x++) {
            for (int y = 0; y < image.getHeight(); y++) {
                Color c = new Color(image.getRGB(x, y));
                int r = c.getRed();
                int g = c.getGreen();
                int b = c.getBlue();
                int a = c.getAlpha();
                if (Math.random()>.5) {
                    r = r & 0xFE;   // ANDed with 0b11111110
                    g = g & 0xFE;   // ANDed with 0b11111110
                    b = b & 0xFE;   // ANDed with 0b11111110
                }
                else {
                    r = r | 0x01;   // ORed with 0b00000001
                    g = g | 0x01;   // ORed with 0b00000001
                    b = b | 0x01;   // ORed with 0b00000001
                }
                Color c2 = new Color(r,g,b);
                image.setRGB(x,y, c2.getRGB());
            }
        }
        return image;
    }
```

To prove this concept is workable, a package was written where BMP files were embedded with another picture. Figure 3 shows the StegoScrubber package that was developed for this task.

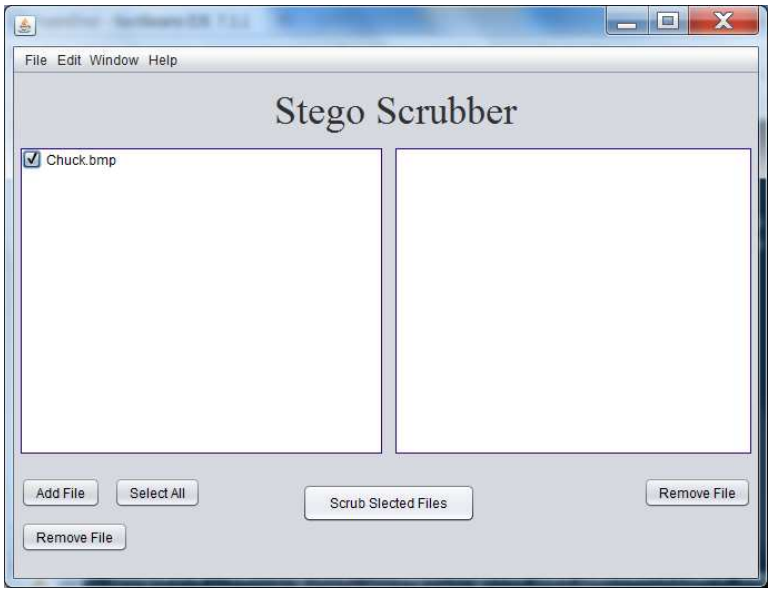

Figure 3 – StegoScrubber removing secret messages from the Chuck.bmp file

A free online software package, *StegHide*, was used to embed a smiley face graphic inside the Chuck.bmp file. After the scrub was complete, the smiley face graphic could no longer be identified. Then the Chuck.bmp file was run 50 more times with no visible degrading of picture quality.

Conclusion

Researchers have rightly determined that hidden content within files is a threat to security. Secret information can be transferred out of secure areas while unwanted content can be transferred into secure areas. Heretofore, the effort has been on detection of secret content, a process called steganalysis.

This design science research makes the contention that staganalytic efforts may be destined to fail, at least in the short run, as newer methods to get around detection are developed. In this article, we illustrate how such a tool can be developed, and develop one that removes secret content embedded through an off-the-shelf steganography package.

REFERENCES

Bahi, J. M., Couchot, J. F., & Guyeux, C. (2012). Steganography: a class of secure and robust algorithms. *The Computer Journal*, 55(6), 653-666.

Cachin, C. (1998) An information-theoretic model for steganography. Information Hiding, Second International Workshop, Portland, Oregon, USA, Berlin, April, Lecture Notes in Computer Science, 1525, 306–318.

Cayre, F., Fontaine, C., & Furon, T. (2005). Watermarking security: Theory and practice. *IEEE Transactions on Signal Processing*, 53(10), 3976-3987.

Comesaña, P., & Pérez-González, F. (2007, September). On a watermarking scheme in the logarithmic domain and its perceptual advantages. In *IEEE International Conference on Image Processing, 2007.* (Vol. 2, pp. II-145).

Dabeer, O., Sullivan, K., Madhow, U., Chandrasekaran, S., & Manjunath, B. S. (2004). Detection of hiding in the least significant bit. *IEEE Transactions on Signal Processing, 52*(10), 3046-3058.

Filler, T., Judas, J., & Fridrich, J. (2011). Minimizing additive distortion in steganography using syndrome-trellis codes. *IEEE Transactions on Information Forensics and Security*, 6(3), 920-935.

Fridrich, J., & Goljan, M. (2002, April). Practical steganalysis of digital images: state of the art. *Electronic Imaging*, 1-13.

Fridrich, J., Goljan, M., & Du, R. (2001). Detecting LSB steganography in color and gray-scale images. *IEEE Multimedia,* 8(4), 22-28.

Fridrich, J., Goljan, M., & Soukal, D. (2003) Higher-order statistical steganalysis of palette images. In *Security and Watermarking of Multimedia Contents* V, E. J. Delp III and P. W. Wong, eds., Proc. SPIE 5020, 178–190.

Hevner, Alan R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS quarterly*, 28(1), 75-105.

Johnson, N. F., & Jajodia, S. (1998). Exploring steganography: Seeing the unseen. *IEEE Computer,* 31(2), 26-34.

Kessler, G. C. (2004). An overview of steganography for the computer forensics examiner. *Forensic Science Communications*, 6(3), 1-27.

Ker, A. D., Bas, P., Böhme, R., Cogranne, R., Craver, S., Filler, T., Fridrich, J., & Pevný, T. (2013, June). Moving steganography and steganalysis from the laboratory into the real world. In *Proceedings of the first ACM workshop on Information hiding and multimedia security*, 45-58.

Kawaguchi, E., & Eason, R. O. (1998). Principles and applications of BPCS steganography. In *Multimedia Systems and Applications*, vol. 3528, 464-473.

Lee, J., Wyner, G. M., & Pentland, B. T. (2008). Process grammar as a tool for business process design. *MIS Quarterly*, 757-778.

Luo, X., Liu, F., & Lu, P. (2007). A LsB steganography approach against pixels sample pairs steganalysis. *International Journal of Innovative Computing, Information and Control*, *3*(3), 575-588.

March, S. T., & Storey, V. C. (2008). Design science in the information systems discipline: an introduction to the special issue on design science research. *Management Information Systems Quarterly*, 32(4), 725-730.

Pevný, T., & Fridrich, J. (2007). Merging Markov and DCT features for multi-class JPEG steganalysis. *Proceedings SPIE, Electronic Imaging, Security, Steganography, and Watermarking of Multimedia Contents IX*, *6505*, 3.

Pevný, T., Bas, P., & Fridrich, J. (2010). Steganalysis by subtractive pixel adjacency matrix. *IEEE Transactions on Information Forensics and Security,* 5(2), 215-224.

Philip, N. S., & Joseph, K. B. (2000). Chaos for Stream Cipher. *Proc. of Recent Advances in Computing and Communications,* McGraw-Hill, 35-42.

Provos, N. (2001), Defending Against Statistical Steganalysis. *10th USENIX Security Symposium*, Washington, DC.

Provos, N., & Honeyman, P. (2003). Hide and seek: An introduction to steganography. *IEEE Security & Privacy*, *1*(3), 32-44.

Solanki, K., Sarkar, A., & Manjunath, B. S. (2007, June). YASS: Yet another steganographic scheme that resists blind steganalysis. In the proceedings of *Information Hiding, 9th International Workshop*, Saint Malo, France, Springer-Verlag, New York, 16-31.

Wang, S., Yang, B., & Niu, X. (2010). A Secure Steganography Method Based on Genetic Algorithm. *Journal of Information Hiding and Multimedia Signal Processing,* 1(1), 28-35.

Westfeld, A., & Pfitzmann, A. (1999). Attacks on Steganographic Systems. In *Information Hiding: Third International Workshop, Dresden, Germany, September 29-October 1, 1999 Proceedings* (No. 1768). Springer Berlin Heidelberg, 61-78.