A Comparative Analysis of Steganographic Tools

Abbas Cheddad, Joan Condell, Kevin Curran and Paul McKevitt

School of Computing and Intelligent Systems, Faculty of Engineering
University of Ulster. Londonderry, Northern Ireland, United Kingdom
Emails: {cheddad-a, j.condell, kj.curran, p.McKevitt}@ulster.ac.uk

Abstract

Steganography is the art and science of hiding data in a transmission medium. It is a sub-discipline of security systems. In this paper we present a study carried out to compare the performance of some common Steganographic tools distributed online. We focus our analysis on systems that use digital images as transmission carriers. A number of these systems exceptionally do not support embedding images rather they allow text embedding; therefore, we constrained the tools to those which embed images files. Visual inspection and statistical comparison methods are the main performance measurements we rely on. This study is an introductory part of a bigger research project aimed at introducing a robust and high payload Steganographic algorithm.

Keywords: Steganography, Image Processing, Security Systems, Statistics.

1 Introduction

In the realm of this digital world Steganography has created an atmosphere of corporate vigilance that has spawned various interesting applications. Contemporary information hiding is due to the author Simmons [Simmons, 1984] for his article titled "The prisoners' Problem and the Subliminal Channel". More recently Kurak and McHugh [Kurak and McHugh, 1992] published their work which resembles embedding into the 4LSBs (Least Significant Bits). They discussed image downgrading and contamination which is known now as Steganography. Steganography is employed in various useful applications e.g., copyright control of materials, enhancing robustness of image search engines and Smart IDs where individuals' details are embedded in their photographs. Other applications are video-audio synchronization, companies' safe circulation of secret data, TV broadcasting, Transmission Control Protocol and Internet Protocol (TCP/IP) packets [Johnson and Jajodia, 1998], embedding Checksum [Bender et al., 2000]…etc. In a very interesting way Petitcolas [Petitcolas, 2000] demonstrated some contemporary applications. One of these was in *Medical Imaging Systems* where a separation is considered necessary for confidentiality between patients' image data or DNA sequences and their captions e.g., Physician, Patient's name, address and other particulars. A link however, must be maintained between the two. Thus, embedding the patient's information in the image could be a useful safety measure and helps in solving such problems. For the sake of providing a fair evaluation of the selected software tools, we restricted our experiments to embedding images rather than text.

The location of the message in the image can vary. The message may be spread evenly over the entire image or may be introduced into areas where it may be difficult to detect a small change such as a complex portion in the image. A complex area is also known as an area of high frequency in which there are considerable changes in colour intensity. Embedding can be performed in the image spatial domain or in the frequency domain. Embedding in the spatial domain can be achieved through altering the least significant bits of the bytes of image pixel values. This process can be in a sequential fashion or in a randomised form. Algorithms based on this method have a high payload, however the method is fragile, prone to statistical attacks and sometimes visual attacks can suffice. The second type of

method, the frequency domain method, is based on the embedding in the coefficient in the frequency domain (i.e., Discrete Cosine Transformation (DCT), Discrete Wavelet Transformation (DWT)). This type of technique is more robust with regard to common image processing operations and lossy compression**.** Another type of method is that of adaptive Steganography which adapts the message embedding technique to the actual content and features of the image. These methods can for example avoid areas of uniform colour and select pixels with large local standard deviation. Edge embedding can also be used alongside adaptive Steganography.

2. Steganographic Tools

The tools that we used for this study are detailed and discussed below. Sources are shown as necessary.

2.1 Hide and Seek (V. 4.1)

Hide and Seek is one of the older methods of Steganography [Wayner, 2002]. It uses a common approach and is relatively easy to apply in image and audio files [Johnson and Katzenbeisser, 2000]. Steganography by this method is carried out by taking the low order bit of each pixel and using it to encode one bit of a character. It creates some noise in the image unless a greyscale image is used. When using Hide and Seek with colour GIFs noise is very obvious. Although it has been asserted that greyscale GIFs do not display any of the artefacts or bad image effects associated with 8-bit colour images which have undergone Steganography, our experiment shows an obvious random salt and pepper like noise on the cover image.

Hide and Seek can be used on 8-bit colour or 8-bit black and white GIF files that are 320 by 480 pixels in size (the standard size of the oldest GIF format) [Wayner, 2002]. There are 19200 (320*480/8) bytes of space available in this GIF image which gets rounded down in practice to 19000 for safe dispersion. In version 4.1 if the cover image is larger than allowed the stego-image is cropped or cut to fit the required size [Johnson et al, 2001]. When an image contains a message it should not be resized because if it has to be reduced part of the message bits will be lost. If the image is too small it is padded with black space. There is also a version 5.0. It works with a wider range of image sizes. However, this version of Hide and Seek also uses a restricted range of image sizes. The images must fit to one of these sizes exactly (320* 200, 320* 400, 320* 480, 640* 400 and 1024* 768) [Johnson et al, 2001]. In version 5 if the image exceeds the maximum allowed which is 1024*768 an error message is returned. If the image is smaller than the minimum size necessary the image containing the message is padded out with black space. The padded areas are added before the message is embedded and are therefore also used as areas in which to hide the message [Johnson et al, 2001]. But if the padded area is removed the message cannot be recovered fully. These characteristics of Hide and Seek stego-images lead searchers/crackers to the fact that a hidden message exists. Hide and Seek 1.0 for Windows 95 has no size limit restrictions and uses an improved technique for information hiding however it can still only be used on 8-bit images with 256 colours or greyscale [Johnson et al, 2001]. BMP images are used with this version instead of GIF images because of licensing issues with GIF image compression [Johnson et al, 2001].

A user chosen key can be inserted into a pseudo random number generator which will determine random number which indicate bytes in the image where the least significant bit is to be changed [Wayner, 2002]. This makes the system more secure as it has two layers of security. The positions in which the message bits are hidden are not in fact random but do follow some sort of pattern. An 8-byte header on the message controls how the message data is dispersed. The first two bytes indicate the length of the message. The second two are a random number key. The key is chosen at random when the message is inserted into the image. The key is firstly inserted into the random number generator [Wayner, 2002]. In Hide and Seek 4.1 there is a built in C code random number generator. A cryptographically secure random number generator could also be used to increase security or IDEA could be used to encrypt the random numbers using a special key. The third pair of bytes is the version of Hide and Seek used. The fourth pair of bytes is used to complete the eight byte block which is necessary for the IDEA cipher [Wayner, 2002]. The 8-byte block is encrypted using the IDEA cipher

which has an optional key and is then stored in the first 8 bytes of the image. If the key is not known the header information cannot be understood and the dispersion of the data in the image cannot be found.

Stego-images will have different properties depending on the version of Hide and Seek used. In version 4.1 and version 5 all palette entries in 256 colour images are divisible by four for all bit values [Johnson et al, 2001]. Greyscale stego-images have 256 triples. They range in sets of four triples from 0 to 252 with incremental steps of 4 (0, 4, 8,…, 248, 252). This can be detected by looking at the whitish value which is 252 252 252. This signature is unique to Hide and Seek [Johnson et al, 2001], [Johnson and Jajodia, 1998]. Later versions of Hide and Seek do not produce the same predictable type of palette patterns as versions 4.1 and 5.0 [Johnson et al, 2001**;** Johnson and Jajodia, 1998].

The DOS command for the Hide and Seek software is as follows:

- `hide <infile.ext> <Cover.gif> [key]`
- `seek <Stego.gif> <outfile.ext> [key]`

2.2 S-Tools (V. 4)

The S-Tools package was written by Andy Brown [Wayner, 2002]. Version 4 can process image or sound files using a single program (S-TOOLS.EXE). S-Tools involves changing the least significant bit of each of the three colours in a pixel in a 24-bit image [Wayner, 2002] for example a 24-bit BMP file [Johnson et al, 2001]. The problem with 24-bit images is that they are not commonly used on the web and tend to stand out (unlike GIF, JPEG, and PNG). This feature is not helpful to Steganography. It involves a pre-processing step to reduce the number of colour entries by using a distance measurement to identify neighbour colours in terms of intensity. After this stage each colour of the dithered image would be associated with two palette entries one of which will carry the hidden data. The software for S-Tools can reduce the number of colours in the image to 256 [Wayner, 2002]. The software uses the algorithm developed by Heckbert [Heckbert, 1982] to reduce the number of colours in an image in a way that will not visually disrupt the image [Wayner, 2002; Martin et al, 2005]. The algorithm plots all the colours in three dimensions (RGB). It searches for a collection of *n* boxes, which contains all of the colours in one of the boxes. The process starts with the complete 256*256*256 space as one box. The boxes are then recursively subdivided by splitting them in the best possible way [Wayner, 2002]. Splitting continues until there are *n* boxes representing the space. When it is finished the programme chooses one colour to represent all the colours in each box. The colour may be chosen in different ways: the centre of the box, the average box colour or the average of the pixels in the box. S-Tools as well as other tools based on LSBs in the spatial domain take for granted that least significant bits of image data are uncorrelated noise [Westfield and Pfitzmann, 1999]. The system interface is easy to use. It supports a drag and drop method to load images. Once the cover image is dragged in; the system will advise the user on how much data in bytes the image can hold.

2.3 Stella (V. 1.0)

The Stella program derives its name from the ***St****eganography* ***Expl****oration* ***La****b,* which is located at the University of Rostock. Its embedding process exploits the visually low prioritised chrominance channels; the YUV-colour system is used. Here, the embedding algorithm considers only one channel and works as follows:

1. Consider the chrominance value of a given pixel.
2. Read a bit from *the secret message*.
3. To embed a "0", decrease the chrominance value of the pixel by *one*.
4. To embed a "1", increase the chrominance value of the pixel by *one*.
5. Go to the next pixel.

It can be assumed that these slight changes of the chrominance values are smaller than a given *JND (Just Noticeable Difference)*.

2.4 Hide in Picture (HIP)

HIP (v 2.1) was created by Davi Tassinari de Figueiredo in 2002. *Hide In Picture* (HIP) uses bitmap images. If the file to be hidden is large, it may be necessary to modify more than a single bit (LSB) from each byte of the image, which can make this difference more visible. With 8-bit pictures, the process is a little more complicated, because the bytes in the picture do not represent colour intensities, but entries in the palette (a table of at most 256 different colours). HIP chooses the nearest colour in the palette whose index contains the appropriate least-significant bits. The HIP header (containing information for the hidden file, such as its size and filename) and the file to be hidden are encrypted with an encryption algorithm, using the password given, before being written in the picture. Their bits are not written in a linear fashion; HIP uses a pseudo-random number generator to choose the place to write each bit. The values given by the pseudo-random number generator depend on your password, so it is not possible for someone trying to read your secret data to get the hidden file (not even the encrypted version) without knowing the password.

2.5 Revelation

Revelation was launched in 2005 by Sean Hamlin. It was entirely coded in Java, developed in the Eclipse IDE. It operates in the same manner as the previous methods in terms of LSB embedding. The basic logic behind their technique is the matching LSB coding which leaves a gray value not altered if its LSB matches the bit to be hidden. Otherwise a colour indexed as $2i$ will be changed to $2i+1$ if the embedded bit is 1, or $2i+1$ is shifted back to $2i$ in case of embedding a 0. Although the software's author claimed the use of a smart embedding method and the Minimum Error Replacement (MER) algorithm to obtain a more natural Stego Image, the latter is prone to first order statistical attack as shown in Figure 3.

3 Steganalysis

There are two stages involved in breaking a Steganographic system: detecting that Steganography has been used and reading the embedded message [Zollner et al, 1998]. Steganalysis methods should be used by the Steganographer in order to determine whether a message is secure and consequently whether a Steganographic process has been successful. Statistical attacks can be carried out using automated methods. A stego-image should have the same statistical characteristics as the carrier so that the use of a stenographic algorithm can not be detected [Westfield and Pfitzmann, 1999]. Therefore a potential message can be read from both the stego-image and the carrier and the message should not be statistically different from a potential message read from a carrier [Westfield and Pfitzmann, 1999]. If it were statistically different the Steganographic system would be insecure. Automation can be used to investigate pixel neighbourhoods and determine if an outstanding pixel is common to the image, follows some sort of pattern or resembles noise. A knowledge base of predictable patterns can be compiled and this can assist in automating the detection process [Johnson and Jajodia, 1998]. Steganalysis tools can determine the existence of hidden messages and even the tools used for embedding. Attacks on Steganography can involve detection and/or destruction of the embedded message. A multiple tools have been introduced to perform Steganalysis; among them is Chi-Square Statistical test, ANOVA test, StegSpy[1], StegDetect[2], Higher-level statistical tests[3], etc. The adopted methods of evaluation in this study are as follows:

---

[1] www.spy-hunter.com

[2] The algorithm described in: http://www.citi.umich.edu/u/provos/papers/detecting.pdf
And the software available at: http://www.outguess.org/

[3] http://www.cs.dartmouth.edu/~farid/publications/tr01.html

3.1 Visual Inspection of the Image

This evaluation method is based on visual inspection of the image. The question is how vulnerable the different Steganographic techniques are to detection through visual inspection of the image for telltale distortion.

3.2 Statistical Analysis

There are two main types of statistical analysis methods investigated here for comparative analysis. These are the peak signal-to-noise ratio and image histograms. They are outlined below.

- *Peak-Signal-to-Noise Ratio*

As a performance measurement for image distortion, the well known Peak-Signal-to-Noise Ratio (*PSNR*) which is classified under the difference distortion metrics can be applied on the stego images. It is defined as:

$$PSNR = 10 \log_{10} \left(\frac{C_{\max}^2}{MSE} \right) \tag{1}$$

where *MSE* denotes Mean Square Error given as:

$$MSE = \frac{1}{MN} \sum_{x=1}^{M} \sum_{y=1}^{N} \left(S_{xy} - C_{xy} \right)^2 \tag{2}$$

and $C_{\max}^2$ holds the maximum value in the image, for example:

$$C_{\max}^2 \leq \begin{cases} 1 \text{ in double precision intensity images} \\ 255 \text{ in 8-bit unsigned integer intensity images} \end{cases}$$

x and y are the image coordinates, M and N are the dimensions of the image, S_{xy} is the generated stego image and C_{xy} is the cover image.

PSNR is often expressed on a logarithmic scale in decibels (dB). *PSNR* values falling below 30dB indicate a fairly low quality (i.e., distortion caused by embedding can be obvious); however, a high quality stego should strive for 40dB and above.

- *Image Histogram*

Histograms are graphics commonly used to display data distributions for quantitative variables. In the case of image; these variables or frequencies are the image intensity values. In this study we can trace any abnormalities in the Stego's histogram.

3.3 Comparative Analysis and Results

The following table (Table 1) tabulates different *PSNR* values spawned by aforementioned software applied on the images shown in Figure 1. Figure 2 and Figure 3 show the output of each of the tools and Figure 4 depicts the pair effect that appears on Stego images generated by the Revelation software. The authors suspect that the pair effect was caused by adopting the sequential embedding concept, which creates new close colours to the existing ones or reduces the frequency difference between adjacent colours.

Figure 1: Images used to generate tables 1. (Left to right) *Set A:* Cover image Boat, (321x481) and the secret image Tank, (155x151). *Set B:* Cover image (Lena 320x480), Secret image (77x92).

Table 1: Summary of performance of the different software reported in this study.

Software	PSNR		Visual Inspection
	Set A	***Set B***	
[Hide&Seek]	18.608	22.7408	Very clear grainy noise in the Stego image, which renders it the worst performer in this study.
[Hide-in-Picture]	23.866	28.316	Little noise/accepts only 24-bit bmp files. Creates additional colour palette entries. In this case the original boat image has 32 colours and the generated Stego augmented the number to 256 by creating new colours.
[Stella]	26.769	16.621	Little noise/Works only with 24-bit images
[S-Tools]	**37.775**	**25.208**	**No visual evidence of tamper**
[Revelation]	23.892	24.381	No visual evidence of tamper, but pair effect appears on the histogram of some outputs

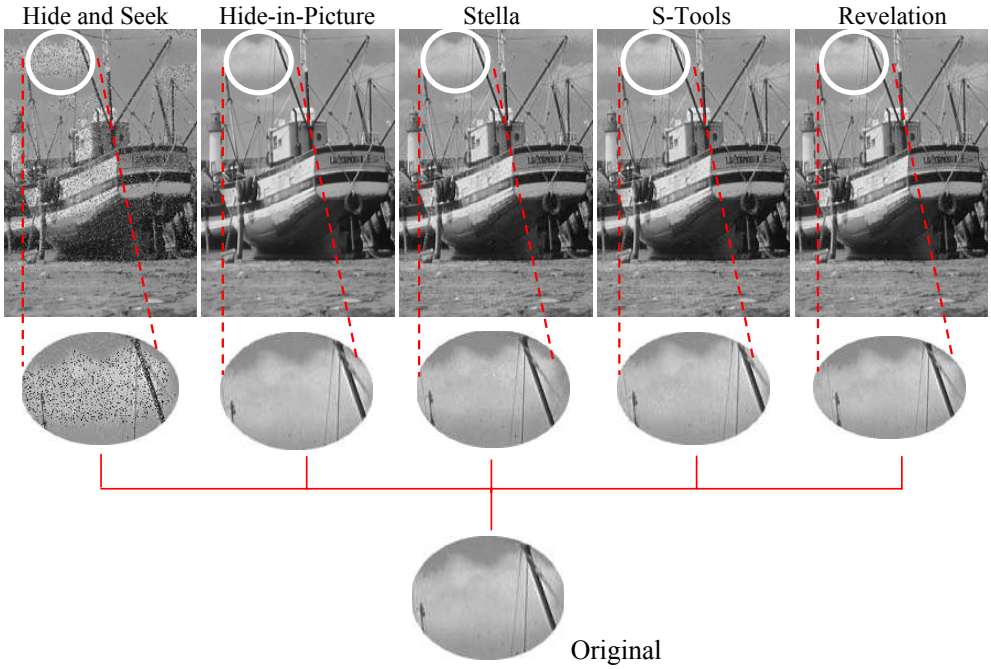

Figure 2: Set A. Stego images of each tool.

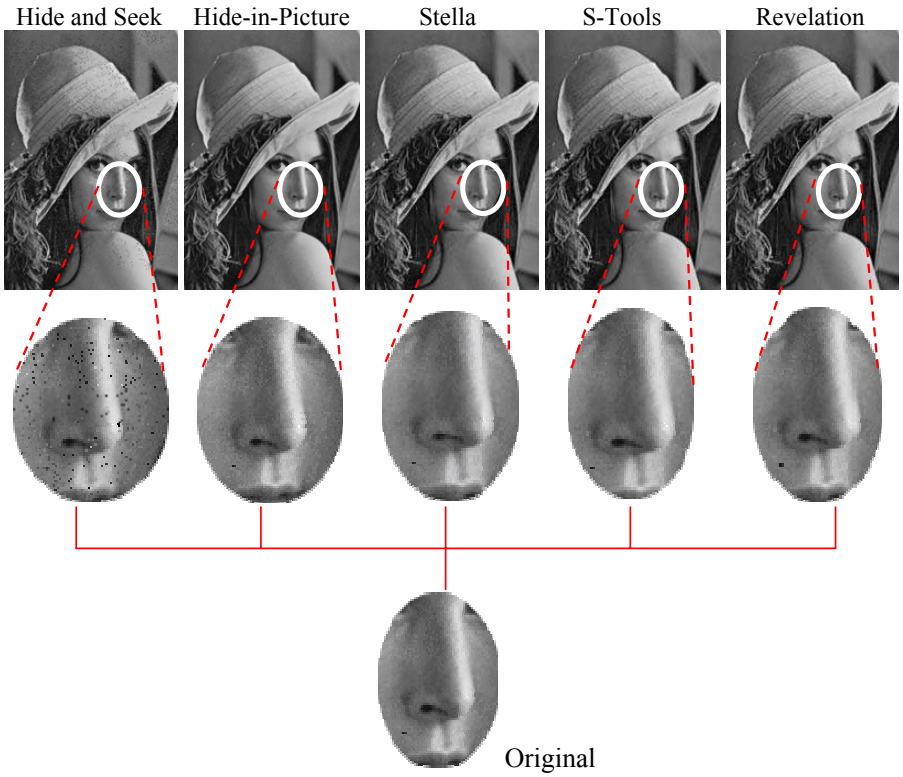

Figure 3: Set B. Stego images of each tool.

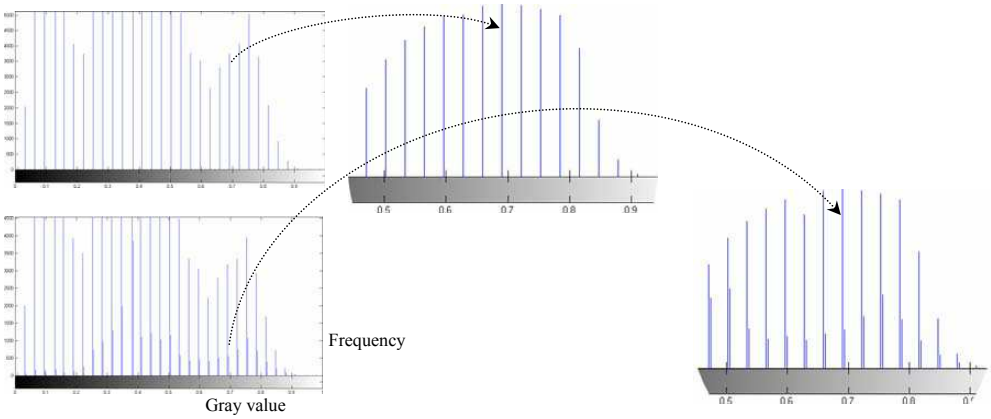

Figure 4: Revelation leaves very obvious pair effect on the histogram. (Top) Original image histogram and (Bottom) Stego image generated by Revelation tool.

4. Conclusion

We have presented a comparative study of some Steganographic tools distributed online. The Revelation tool seems to do a good job in hiding any visual tamper on the cover image, but the histogram of its generated output reveals some traces left by the tool which draws suspicion to the stego image. Hide-In-Picture earned a good PSNR value but the cover image was distorted a bit after the embedding. Hide and Seek shows very clear grainy salt and pepper noise on the stego image, the noise appears random. S-Tools shows better performance taking into consideration the two factors (i.e., PSNR and Visual Inspection). Stella leaves prints at the end of the stego file which can be picked up easily by some Steganalysis software like the one we tried, which is an open source application (*ImageHide Hidden Data Finder v0.2*, see "Internet Resources" at the References section). All the above tools can not resist image compression and that is why all their input image files were of lossless type (i.e., BMP, GIF). The authors affirm that S-Tools algorithm has the highest performance and its software provides a better graphical interface. It should be worth noting here that Steganographic tools have undergone significant improvement by exploiting JPEG compression method. For example F5 and OutGuess algorithms (see references) are strong enough to resist major attacks, moreover their resulting image Stego is of high quality (i.e., high PSNR). This study is meant for evaluating tools that act in the spatial domain, thus discussing F5 and OutGuess is out of scope of this evaluation.

References

[Simmons, 1984] Simmons, G. J., (1984). *The Prisoners' Problem and the Subliminal Channel. Proceedings of CRYPTO83- Advances in Cryptology, August 22-24. 1984. pp. 51.67.*

[Kurak, 1992] Kurak, C. and McHugh, J., (1992). *A cautionary note on image downgrading. Proceedings of the Eighth Annual Computer Security Applications Conference. 30 Nov-4 Dec 1992 pp. 153-159.*

[Johnson, and Jajodia, 1998] Johnson, N. F. and Jajodia, S., (1998). *Exploring Steganography: Seeing the Unseen. IEEE Computer, 31 (2): 26-34, Feb 1998.*

[Bender et al., 2000] Bender, W., Butera, W., Gruhl, D., Hwang, R., Paiz, F.J. and Pogreb, S., (2000). *Applications for Data Hiding. IBM Systems Journal, 39 (3&4): 547-568.*

[Petitcolas, 2000] Petitcolas, F.A.P., (2000). *"Introduction to Information Hiding". In: Katzenbeisser, S and Petitcolas, F.A.P (ed.) (2000) Information hiding Techniques for Steganography and Digital Watermarking. Norwood: Artech House, INC.*

[Wayner, 2002] Wayner, P. (2002). *Disappearing Cryptography*. *2nd ed. USA: Morgan Kaufmann Publishers.*
[Johnson and Katzenbeisser, 2000] Johnson and Katzenbeisser, (2000). *"A survey of Steganographic techniques". In: Katzenbeisser, S and Petitcolas, F.A.P (ed.) (2000) Information hiding Techniques for Steganography and Digital Watermarking. Norwood: Artech House, INC.*
[Johnson et al., 2001] Johnson Neil F., Zoran Duric, Sushil Jajodia, Information Hiding, Steganography and Watermarking - Attacks and Countermeasures, Kluwer Academic Publishers, 2001.
[Heckbert, 1982] Heckbert Paul, Colour Image Quantization for Frame Buffer Display. In Proceedings of SIGGRAPH 82, 1982.
[Martin et al, 2005] Martin, A., Sapiro, G. and Seroussi, G., (2005). *Is Image Steganography natural?. IEEE Trans on Image Processing, 14(12): 2040-2050, December 2005.*
[Westfield and Pfitzmann, 1999] Andreas Westfield and Andreas Pfitzmann, Attacks on Steganographic Systems Breaking the Steganography Utilities EzStego, Jsteg, Steganos and S-Tools and Some Lessons Learned. Dresden University of Technology, Department of Computer Science, Information Hiding, Third International Workshop, IH'99 Dresden Germany, September / October Proceedings, Computer Science 1768. pp. 61- 76, 1999.
[Zollner et al, 1998] Zollner J., H. Federrath, H. Klimant, A. Pfitzmann, R. Piotraschke, A. Westfeld, G. Wicke, G. Wolf, Modelling the Security of Steganographic Systems, Information Hiding, Second International Workshop, IH'98 Portland, Oregon, USA, Proceedings, Computer Science 1525. pp. 344-354, April 1998.

Internet Resources

[Hide and Seek]:
ftp://ftp.funet.fi/pub/crypt/mirrors/idea.sec.dsi.unimi.it/cypherpunks/steganography/hdsk41b.zip
[S-Tools]: ftp://ftp.funet.fi/pub/crypt/mirrors/idea.sec.dsi.unimi.it/code/s-tools4.zip
[Stella]: http://wwwicg.informatik.uni-rostock.de/~sanction/stella/
[Hide in Picture]: http://sourceforge.net/projects/hide-in-picture/
[Revelation]: http://revelation.atspace.biz/
[ImageHide Hidden Data Finder]: http://www.guillermito2.net
[F5]: http://wwwrn.inf.tu-dresden.de/~westfeld/f5.html
[OutGuess]: http://www.outguess.org/