Attacks on Steganographic Systems

Breaking the Steganographic Utilities EzStego, Jsteg, Steganos, and S-Tools—and Some Lessons Learned

Andreas Westfeld and Andreas Pfitzmann

Dresden University of Technology
Department of Computer Science
D-01062 Dresden, Germany
{westfeld, pfitza}@inf.tu-dresden.de

Abstract. The majority of steganographic utilities for the camouflage of confidential communication suffers from fundamental weaknesses. On the way to more secure steganographic algorithms, the development of attacks is essential to assess security. We present both *visual attacks*, making use of the ability of humans to clearly discern between noise and visual patterns, and *statistical attacks* which are much easier to automate.
The visual attacks presented here exemplify that at least EzStego v2.0b3, Jsteg v4, Steganos v1.5, and S-Tools v4.0 suffer from the misassumption that least significant bits of image data are uncorrelated noise. Beyond that, this paper introduces more objective methods to detect steganography by statistical means.

1 Introduction

Steganography is no routine means to protect confidentiality. Normally, cryptography is used to communicate confidentially. Cryptographic algorithms—the security of which can be proven or traced back to known hard mathematical problems—are widely available. However, in contrast to steganography, cryptographic algorithms generate messages which are recognisable as encrypted messages, although their content remains confidential.

Steganography[1] embeds a confidential message into another, more extensive message which serves as a carrier. The goal is to modify the carrier in an imperceptible way only, so that it reveals nothing—neither the embedding of a message nor the embedded message itself.

The functioning of a steganographic system is shown in Fig. 1: The sender creates a steganogram using the embedding function which function has two parameters:

1. a carrier medium containing randomness (e. g., noise), and
2. the message to be embedded.

[1] στεγανός + γράφειν, covered writing

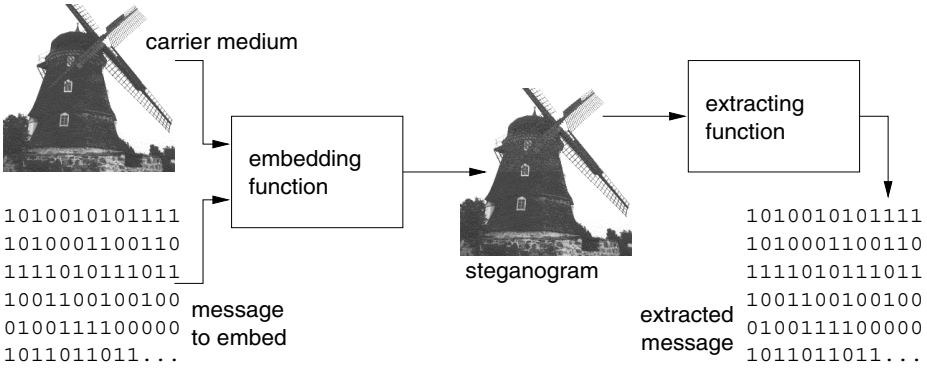


Fig. 1. Steganographic system

Multimedia data, such as audio and video, are excellent carriers. After digitisation, they contain so-called quantisation noise which provides space to embed data. Lossy compression may introduce another kind of noise. Using the extracting function, the recipient must be able to reproduce the embedded message from the steganogram.

A steganogram should have the same statistical characteristics as the carrier media so that the use of a steganographic algorithm can not be detected. Consequently, a (potential) message can be read from both the steganogram and the carrier medium. A message read from a steganogram must not be statistically different from a potential message read from a carrier medium—otherwise, the steganographic system would be insecure.

Some steganographic utilities use secret keys. We can distinguish two kinds of keys: steganographic keys and cryptographic keys [4]. A steganographic key controls the embedding and extracting process. For example, it can scatter the message to be embedded over a subset of all suitable places in the carrier medium. Without the key, this subset is unknown, and each sample used to detect embedding by a statistical attack is a mixture of used and unused places (i. e., of all potential places) which spoils the result. A cryptographic key, however, is used to encrypt the message before it is embedded. For both applications the "secret", which conceals the message, is detached from the actual algorithm in the form of a parameter—the key. If the key is confidential, the steganographic algorithm can be public (Kerckhoffs' Principle). It is possible to decide whether the bits read are in fact an encoded message of a potential steganogram only if one has the appropriate decryption key. Encryption is also advisable in addition to steganographic utilities which do not implicitly encrypt.

To decouple the security of steganographic algorithms from the appearance of the hidden message, we use pseudo random bit-strings to generate these messages in our experiments. Such bit-strings have all statistical properties of encrypted messages. In this paper, we will concentrate on images, the most widespread carrier medium.

Related to this work is the Final Year Project of Tinsley [5] on Steganography and JPEG Compression. He describes statistical attacks applied to Jsteg [14] using a different statistical model. Fravia's pages explain brute force attacks to steganography [11]. Finally, there was an introduction to "Steganalysis" given by Johnson at the previous Workshop on Information Hiding in 1998 [2].

In the following sections, we present our attacks on EzStego v2.0b3, Jsteg v4, Steganos v1.5, and S-Tools v4.0, going into details of each utility attacked where needed. To have a fundamental example, we first describe EzStego in Sect. 2. In Sect. 3, we describe our visual attacks. Thereafter, we describe our statistical attacks in Sect. 4. Finally, we present our conclusions and outlook in Sect. 5.

2 EzStego

The utility EzStego (by Romana Machado) embeds messages in GIF files. GIF files [12] contain a colour palette with up to 256 different colours out of 2^{24} possible, and the Lempel-Ziv-Welch (LZW) compressed [3, 6, 8] matrix of palette indices. EzStego embeds messages into the pixels without any length information. It leaves the colour palette unmodified.

The steganographic algorithm creates a sorted copy of the palette. It sorts in a way that we can hardly tell the difference between two adjacent colours in the sorted palette. Sorting by luminance is not optimal in any case because two colours with the same luminance could be radical different. We can interpret each colour as a point in a three-dimensional space, the RGB (red, green, blue) colour cube.

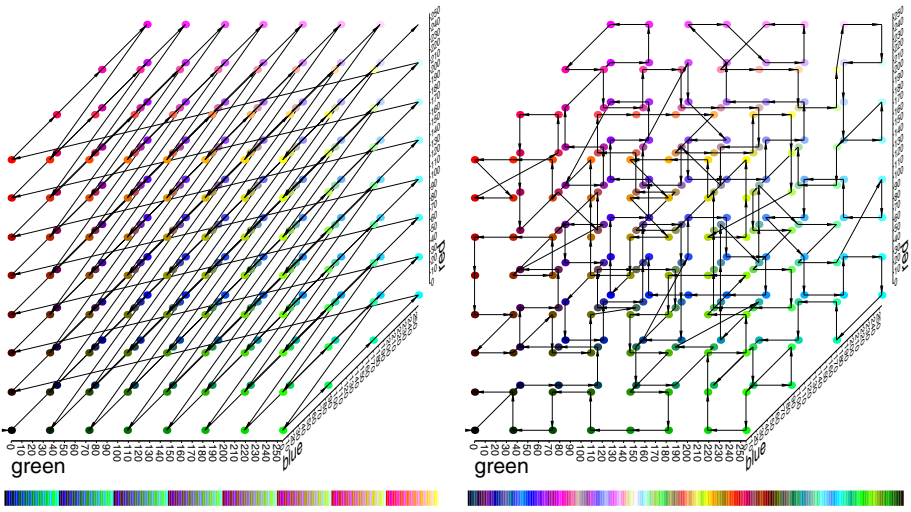


Fig. 2. Colour order in the palette (l.) and sorted as used by EzStego (r.)

Fig. 2 shows the order of colours in the RGB cube. On the left the colours look more sorted than on the right. This is the order of the colours in the palette, in most cases a numerical order. On the right the colours are sorted by EzStego to follow a shortest path through the RGB cube.

The embedding function of EzStego works line by line on the unbroken sequence of pixels from top left to bottom right. After embedding, each pixel holds one steganographic value (i. e., one bit of the embedded message). The steganographic value of a pixel is the least significant bit its index would have in the sorted palette. The embedding function matches the steganographic value with the bit to be embedded (i. e. if the bit to be embedded is not already there), and replaces the colour by its neighbour in the sorted palette if necessary.

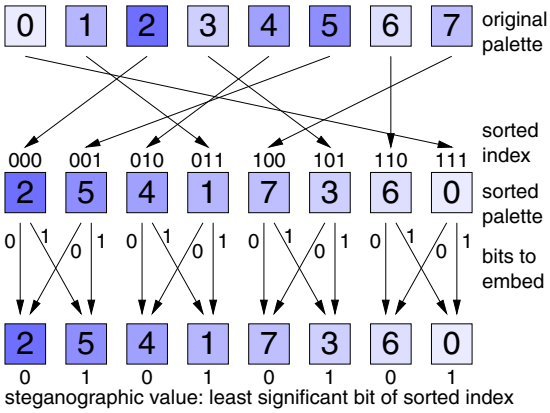


Fig. 3. Embedding function of EzStego

Fig. 3 shows the embedding function of EzStego with a reduced palette. For example, we find index 7 for a given pixel in the carrier image. If we want to embed a '1', we replace the index by 3, and if we want to embed a '0' we change nothing. Because the colour of index 7 in the original palette is at index 100 (=4) in the sorted palette, and the colour of index 3 is at index 101 (=5) in the sorted palette, both colours are neighbours in the sorted palette, i. e. hardly to distinguish. A change from index 7 to index 3 (and vice versa) is imperceptible for our eyes unless we compare it directly with the original image.

Everybody can extract the (imaginary) message bits easily. If there is one embedded bit per pixel we can draw them as an image—e. g. white for the steganographic value '1', and black for the value '0'.

3 Visual Attacks

Independently from each other, several authors assumed that least significant bits of luminance values in digital images are completely random and could

therefore be replaced (references: Contraband [9], EzStego [10], Hide & Seek [13], PGMStealth [15], Piilo [16], Scytale [17], Snow [18], Steganos [19], Stego [20], Stegodos [21], S-Tools [22], White Noise Storm [23]). By the visual attacks described in this section, we will reveal that this assumption is wrong. The majority of steganographic algorithms embeds messages replacing carefully selected bits by message bits. Actually, it is difficult to distinguish randomness and image contents by machine, and it is even more difficult to distinguish least significant bits and random bits. It is extremely difficult to specify permissible image content in a formal way. A substitute is having people realise what image content is. However, the border becomes blurred and depends on our imagination—who did not already detect shapes in a cloud formation? The human sight is trained to recognise known things. This human ability is used for the visual attacks. Fig. 5 represents the least significant bits of Fig. 4, which is actually not an attack on steganography. We still can see the windmill in the least significant bits in both images, and we are not able to identify the steganogram with our eyes, although the upper half of the image on the right contains a steganographic message.

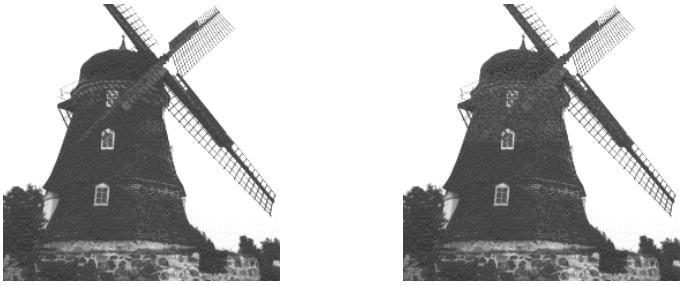


Fig. 4. Windmill as carrier medium (l.), and steganogram (r.)

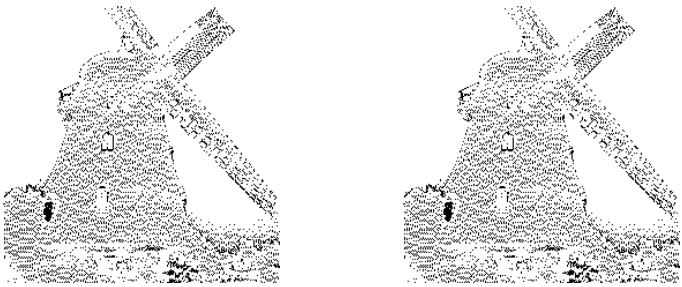
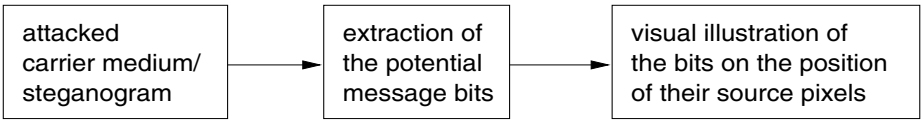


Fig. 5. Least significant bits of the images in Fig. 4, black for LSB=0, white for LSB=1

3.1 The Idea of Visual Attacks

The idea of visual attacks is to remove all parts of the image covering the message. The human eye can now distinguish whether there is a potential message or still image content. The filtering process depends on the presumed steganographic utility, and it has the following structure:



3.2 An Embedding Filter for Visual Attacks

An embedding filter for visual attacks graphically presents the values pixels yield when the extraction function is applied to them. EzStego uses the colours of pixels, defined by the palette, to determine the embedded bits. The embedding filter for visual attacks on EzStego replaces the original palette by a black and white palette. This is depicted in Fig. 6.

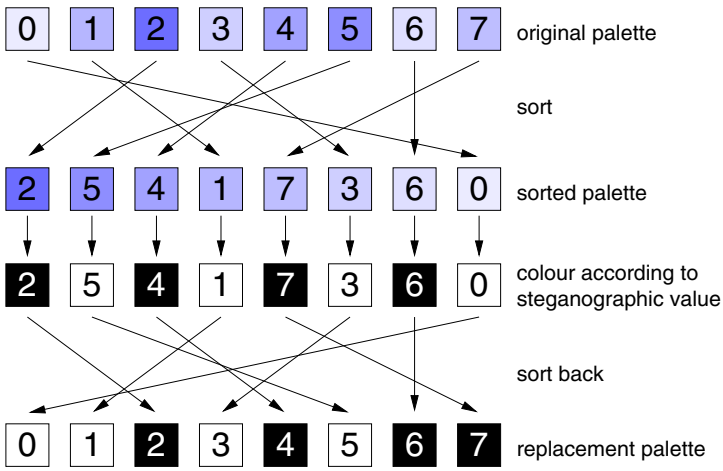


Fig. 6. Assignment function of replacement colours; colours that have an even index in the sorted palette become black, the rest becomes white.

3.3 Experiments

The following examples of visual attacks clearly show the assumption to be a myth that least significant bits are completely random and therefore might be replaced. To produce these examples, we developed small Java applications [24].

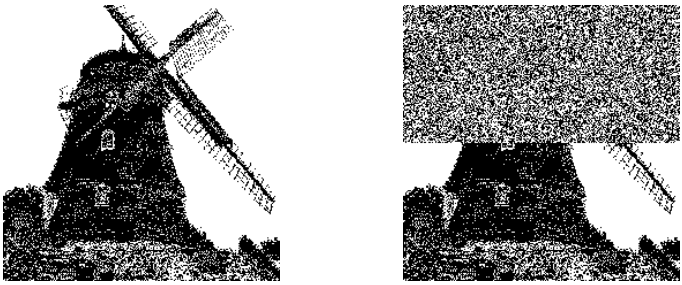


Fig. 7. EzStego; filtered images of Fig. 4: nothing embedded (l.), 50 % capacity of the carrier used for embedding (r.)

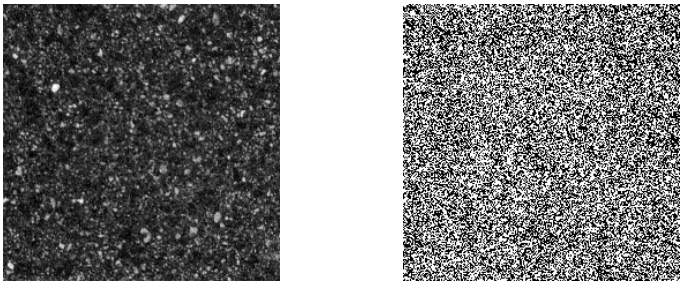


Fig. 8. GIF image of a flooring tile as carrier medium, and its filtered image

EzStego—continuous embedding. Messages, that do not use the maximum length possible, leave the rest of the carrier medium unchanged. EzStego does not encrypt the message contents. It is easy to recognise where the message is in Fig. 7, but it depends on the image content, as Fig. 8 shows. There is no embedded message in the flooring tile image.

S-Tools—spread embedding. The S-Tools spread a message over the whole carrier medium. In contrast to EzStego, there is no clear dividing line between the unchanged rest, left over with shorter messages, and the steganographically changed pixels. Both of them are mixed. In the right images of Fig. 9, Fig. 10, and Fig. 11 there are eight colors, one bit in each of the three colour components, because S-Tools embeds up to three bits per pixel (see [24] for the coloured version).

Steganos—continuous embedding with fill up. Steganos uses the carrier medium completely in every case. It will fill up shorter messages, as shown in Fig. 13. Filtered steganograms never contain content of the initial image (Fig. 12).

Fig. 9. True Colour BMP image as carrier medium, and its filtered image

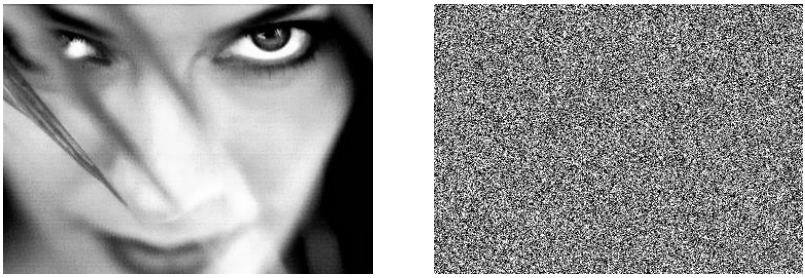


Fig. 10. S-Tools; steganogram with maximum size of embedded text, and its filtered image

Jsteg—embedding in a transformed domain. Jsteg [14] embeds in JPEG images. In JPEG images, the image content is transformed into frequency coefficients to achieve storage as compact as possible. There is no visual attack in the sense presented here, because one steganographic bit influences up to 256 pixels.

4 Statistical Attacks

4.1 Idea of the Chi-square Attack

The embedding function of EzStego overwrites least significant bits of the sorted indices. Overwriting least significant bits transforms values into each other which only differ in the least significant bit. These *pairs of values* are called PoVs in the sequel. If the bits used for overwriting the least significant bits are equally distributed, the frequencies of both values of each PoV become equal. Fig. 14 uses the example of Fig. 3 to show how the frequencies of the colours of a picture are changed, when EzStego is used to embed an equally distributed message. The idea of the statistical attack is to compare the theoretically expected frequency distribution in steganograms with some sample distribution observed in the possibly changed carrier medium.

Fig. 11. S-Tools; steganogram with 50 % capacity of the carrier medium used, and its filtered image

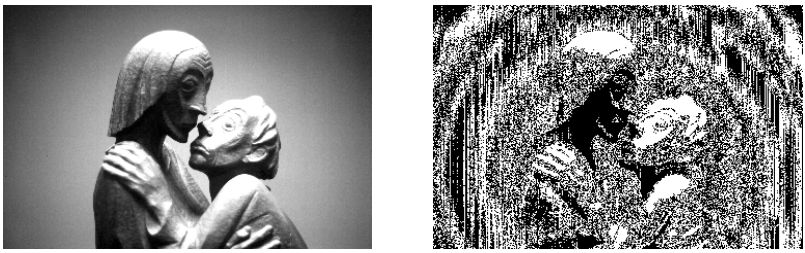


Fig. 12. True Colour BMP image as carrier medium, and its filtered image

A critical point is how to obtain the theoretically expected frequency distribution (i. e., the frequency of occurrence we would expect after applying steganographic changes). This frequency must not be derived from our random sample, because this random sample could have been changed by steganographic operations. But in most cases we don't have the original to compare with or to derive the expected frequency from. In the original, the theoretically expected frequency is the arithmetic mean of the two frequencies in a PoV. The dashed line in Fig. 14 connects these arithmetic mean values. Because the embedding function overwrites the least significant bits, it does not change the sum of these two frequencies. The count taken from the odd value frequency is transferred to the corresponding even value frequency in each PoV, and vice versa. As the sum stays constant, the arithmetic mean is the same for a PoV in both, the original carrier medium and each corresponding steganogram. This fact allows us to obtain the theoretically expected frequency distribution from the random sample. So we don't need the original carrier medium for the attack.

The degree of similarity of the observed sample distribution and the theoretically expected frequency distribution is a measure of the probability that some embedding has taken place. The degree of similarity is determined using the Chi-square test (e.g., [1]). This test operates on a mapping of observations into categories. It performs the following steps:

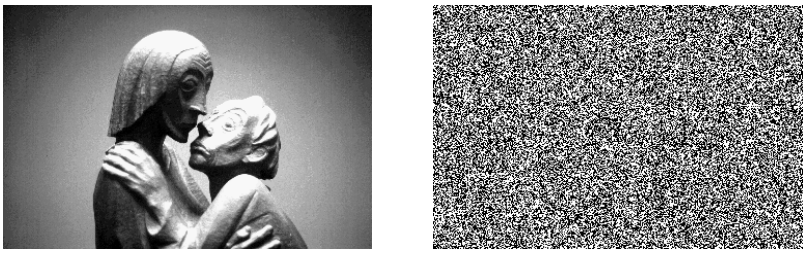


Fig. 13. Steganos; steganogram with only one byte of embedded text, and its filtered image

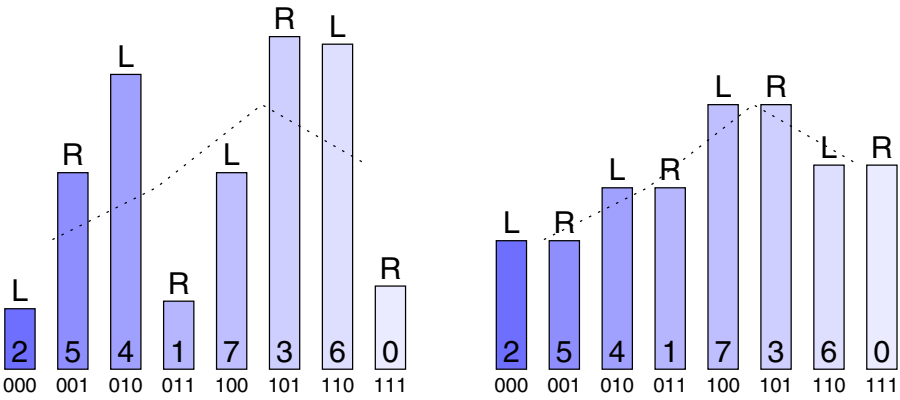


Fig. 14. Histogram of colours before and after embedding a message with EzStego

1. We shall suppose that there are k categories and that we have a random sample of observations. Each observation must fall in one and only one category. The categories are all palette indices, the colour of which is placed at an even index within the sorted palette. Without restricting generality, we concentrate on the odd values of the PoVs of the attacked carrier medium. Their minimum theoretically expected frequency must be greater than 4, we may unify categories to hold this condition.
2. The theoretically expected frequency in category i after embedding an equally distributed message is

$$n_i^* = \frac{|\{\text{colour}|\text{sortedIndexOf(colour)} \in \{2i, 2i+1\}\}|}{2}$$

3. The measured frequency of occurrence in our random sample is

$$n_i = |\{\text{colour}|\text{sortedIndexOf(colour)} = 2i\}|$$

4. The χ^2 statistic is given as $\chi^2_{k-1} = \sum_{i=1}^{k} \frac{(n_i - n_i^*)^2}{n_i^*}$ with $k-1$ degrees of freedom.

5. p is the probability of our statistic under the condition that the distributions of n_i and n_i^* are equal. It is calculated by integration of the density function:

$$p = 1 - \frac{1}{2^{\frac{k-1}{2}}\Gamma(\frac{k-1}{2})} \int_0^{\chi^2_{k-1}} \mathrm{e}^{-\frac{x}{2}} x^{\frac{k-1}{2}-1} \mathrm{d}x \tag{1}$$

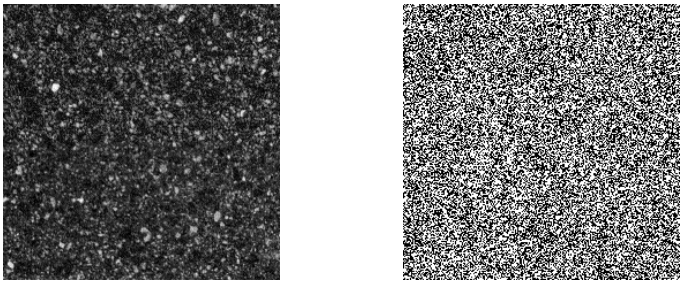


Fig. 15. Flooring tile as steganogram of EzStego, and filtered; this visual attack cannot distinguish between the upper, steganographic half and the lower, original half.

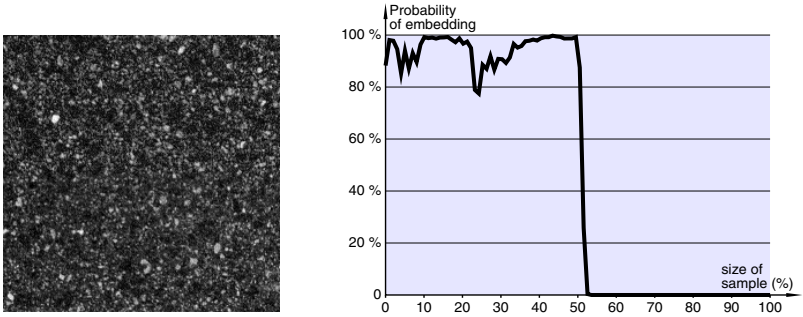


Fig. 16. Probability of embedding with EzStego in the flooring tile image (Fig. 15)

4.2 Experiments

EzStego—continuous embedding. Fig. 15 depicts a steganogram, in which a secret message of 3 600 bytes has been embedded, the same message as in Fig. 4. Fig. 15 looks pretty much like Fig. 8, due to the contents of the picture. The visual attack reaches its limit. The diagram in Fig. 16 presents the p-value of the Chi-square test as a function of an increasing sample. This p-value is roughly the probability of embedding. Initially, the sample comprises 1 % of the pixels,

starting from the upper border. For this sample, Equ. (1) yields a probability of embedding of $p = 0.8826$. The next sample comprises an additional 1 % of the pixels, i. e. 2 % of the whole picture. The p-value increases to 0.9808. As long as the sample comprises pixels of the upper half only, in which has been embedded, the p-value does not drop below 0.77. The pixels of the lower half of the picture are unchanged, because the message to be embedded was not such long. A sample of 52 % of the pixels comprises enough unchanged pixels to let the p-value drop to essentially 0. (Here, "essentially" means that the probability is smaller than the numeric precision of the 80-bit floating point arithmetic used for the implementation.)

S-Tools—spread embedding. The S-Tools spread the embedded bits over the whole carrier medium. Therefore, diagrams of the type of Fig. 16 are not useful for S-Tools. Instead, Table 1 characterises the effectiveness of our statistical test by applying it to some files with nothing embedded, 50 % embedded, or 99.5 % embedded, respectively. Actually this simple test is too weak to detect spreaded changes (see example jungle50.bmp in Table 1). More sensitive tests take appropriate combinations of the k categories or different categories. Some experiments showed useful results with only 33 % of embedded text in colour images, but tests for less embedded text causes ε (which stands for the probability of error in Table 1) to reach 0.5 rapidely.

Steganos—continuous embedding with fill up. Table 2 gives the result of the same experiment on Steganos. If we embed only one byte with Steganos (the shortest message which is possible), we get the same small probability of error as if we use 100 % capacity of the carrier medium. This is due to the fact that the stream cipher used to encrypt the secret message fills up the message with padding bytes until the capacity of the carrier medium is exhausted.

Jsteg—embedding in a transformed domain. As already noted in Sect. 3, visual attacks do not work concerning Jsteg. Since Jsteg (as EzStego) embeds bits continuously, we use the former presentation of Fig. 16 in Fig. 17, Fig. 18, and Fig. 19. They show that our statistical test is quite effective concerning Jsteg as well.

5 Conclusions and Outlook

The embedding strategy of most stego-systems which overwrite least significant bits of the carrier medium, withstands at most the casual unsophisticated observer:

- The visual attacks described show that in pictures, least significant bits are not completely random, but bare a correlation with each other clearly discernible by the human sight if the pictures are presented using an embedding filter for visual attacks described above.

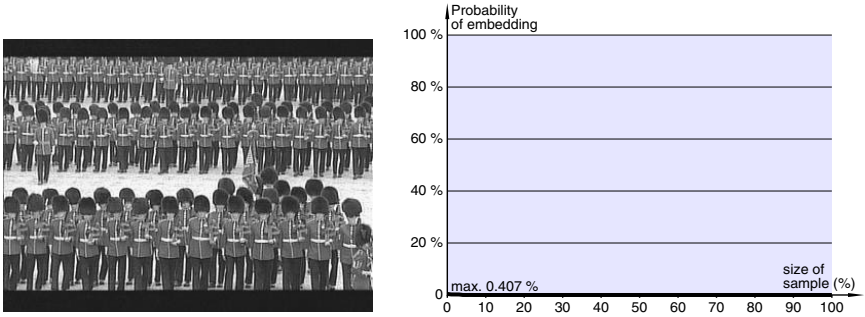


Fig. 17. JPEG image as carrier medium; nothing is embedded, and the statistical test yields a very low probability of embedding

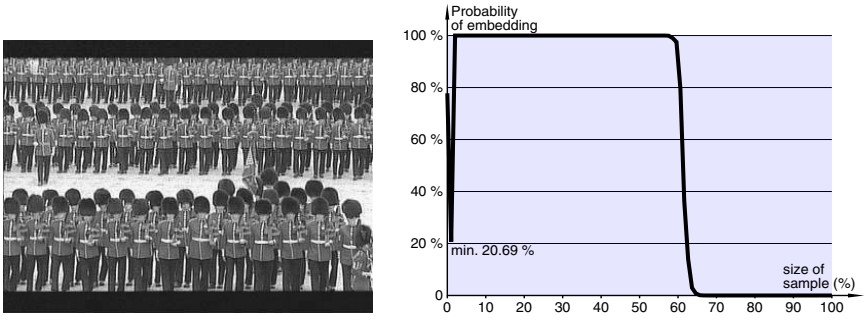


Fig. 18. Jsteg; steganogram with 50 % embedded

– Overwriting least significant bits equals frequencies of occurrence which would be unequal otherwise with very high probability. Using statistical tests, this equalisation can clearly be detected—as we have shown.

Where available, statistical tests are superior to visual attacks: They are less dependent on the cover used and they can be fully automated and thereby applied on a large scale.

By not overwriting all least significant bits, but only a fraction of them and by choosing these bits (pseudo)randomly, the error rate both of the visual and statistical attacks increases. But by that measure, the throughput of the steganographic system decreases. In the limiting case, we have a steganographic system which is nearly undetectable, but which transmits nearly nothing.

The following alternatives are promising, but need validation by a hopefully "hostile" stego-community as well:

– We should concentrate the embedding process exclusively on the randomness in the carrier medium. Of course, it is all but trivial to find out what is completely random within a carrier. [7] is an example how to design a steganographic system that way.

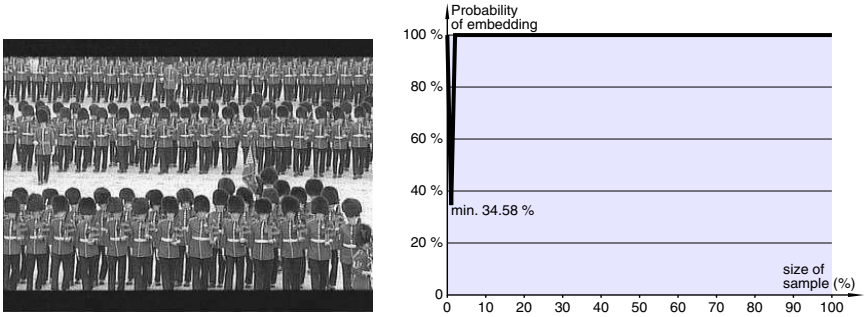


Fig. 19. Jsteg; steganogram with maximum size of embedded text

Table 1. Probability of embedding for S-Tools

file	size of embedded text	p-value
jungle.bmp	0	$0+\varepsilon$
bavarian.bmp	0	$0+\varepsilon$
soccer.bmp	0	$0+\varepsilon$
groenemeyer.bmp	0	$0+\varepsilon$
pudding.bmp	0	$0+\varepsilon$
jungle50.bmp	18 090 bytes/50 %	$0+\varepsilon$
jungle100.bmp	36 000 bytes/99.5 %	$1-\varepsilon$
bavarian100.bmp	36 000 bytes/99.5 %	$1-\varepsilon$
soccer100.bmp	36 000 bytes/99.5 %	$1-\varepsilon$
groenemeyer100.bmp	36 000 bytes/99.5 %	$1-\varepsilon$
		$\varepsilon < 10^{-16}$

– We should replace the operation *overwrite* by other operations (e. g., by *increment*). Then the frequencies are not balanced, but circulate in the range of values.

Cryptography gained the security of today's state-of-the-art systems by an iterative process of designing and publishing cryptosystems, analysing and breaking them, and then re-designing hopefully more secure ones—and exposing them once more to attacks. This iterative process has to take place concerning steganography as well. Since steganography has at least one more parameter than cryptography, the choosing of cover within a carrier, validation is more complex and may take longer and proofs of security (if any) are even more limited than concerning cryptography. Within the validation circle of steganographic systems, this paper is—hopefully—a step forward.

Table 2. Probability of embedding for Steganos

file	size of embedded text	p-value
army.bmp	0	0.0095887
bavarian.bmp	0	$0 + \varepsilon$
soccer.bmp	0	$0 + \varepsilon$
groenemeyer.bmp	0	$0 + \varepsilon$
pudding.bmp	0	$0 + \varepsilon$
army100.bmp	12 000 bytes/99.5 %	$1 - \varepsilon$
bavarian1.bmp	1 byte/0.008 %	$1 - \varepsilon$
soccer1.bmp	1 byte/0.008 %	$1 - \varepsilon$
groenemeyer1.bmp	1 byte/0.008 %	$1 - \varepsilon$
pudding1.bmp	1 byte/0.008 %	$1 - \varepsilon$
		$\varepsilon < 10^{-16}$

References

1. Wilfrid J. Dixon, Frank J. Massey: Introduction to Statistical Analysis. McGraw-Hill Book Company, Inc., New York 1957.
2. Neil F. Johnson, Sushil Jajodia: Steganalysis of Images Created Using Current Steganography Software, in David Aucsmith (Ed.): Information Hiding, LNCS 1525, Springer-Verlag Berlin Heidelberg 1998. pp. 32–47
3. M. R. Nelson: LZW Data Compression. Dr. Dobb's Journal, October 1989.
4. Birgit Pfitzmann, Information Hiding Terminology, in Ross Anderson (Ed.): Information Hiding. First International Workshop, LNCS 1174, Springer-Verlag Berlin Heidelberg 1996. pp. 347–350
5. Robert Tinsley, Steganography and JPEG Compression, Final Year Project Report, University of Warwick, 1996
6. Terry Welch: A Technique for High-Performance Data Compression. IEEE Computer, June 1984.
7. Andreas Westfeld, Gritta Wolf: Steganography in a Video Conferencing System, in David Aucsmith (Ed.): Information Hiding, LNCS 1525, Springer-Verlag Berlin Heidelberg 1998. pp. 32–47
8. Jacob Ziv, Abraham Lempel: A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory, May 1977.

Internet Sources

9. Contraband, `http://www.galaxycorp.com/009`
10. EzStego, `http://www.fqa.com/romana/`
11. Fravia's Steganography, `http://www.fravia.org/stego.htm`
12. GIF, `http://members.aol.com/royalef/gif89a.txt`
13. Hide and Seek, `http://www.rugeley.demon.co.uk/security/hdsk50.zip`
14. Jsteg, `ftp://ftp.funet.fi/pub/crypt/steganography/`
15. PGMStealth, `http://www.sevenlocks.com/security/SWSteganography.htm`
16. Piilo, `ftp://ftp.funet.fi/pub/crypt/steganography/`
17. Scytale, `http://www.geocities.com/SiliconValley/Heights/5428/`
18. Snow, `http://www.cs.mu.oz.au/~mkwan/snow/`

19. Steganos, http://www.demcom.com/deutsch/index.htm
20. Stego, http://www.best.com/~fqa/romana/romanasoft/stego.html
21. Stegodos, http://www.netlink.co.uk/users/hassop/pgp/stegodos.zip
22. S-Tools, ftp://idea.sec.dsi.unimi.it/pub/security/crypt/code/s-tools4.zip
23. White Noise Storm, ftp://ftp.funet.fi/pub/crypt/mirrors/idea.sec.dsi.unimi.it/cypherpunks/steganography/wns210.zip
24. http://wwwrn.inf.tu-dresden.de/~westfeld/attacks.html