Steganalysis of JPEG Images: Breaking the F5 Algorithm

Jessica Fridrich[1], Miroslav Goljan[1], Dorin Hogea[2]

[1] Department of Electrical and Computer Engineering, SUNY Binghamton, Binghamton,
NY 13902-6000, USA
{fridrich,mgoljan}@binghamton.edu
http://www.ssie.binghamton.edu/fridrich
[2] Department of Computer Science, SUNY Binghamton, Binghamton, NY 13902-6000,
USA
dhogea1@binghamton.edu

Abstract. In this paper, we present a steganalytic method that can reliably detect messages (and estimate their size) hidden in JPEG images using the steganographic algorithm F5. The key element of the method is estimation of the cover-image histogram from the stego-image. This is done by decompressing the stego-image, cropping it by four pixels in both directions to remove the quantization in the frequency domain, and recompressing it using the same quality factor as the stego-image. The number of relative changes introduced by F5 is determined using the least square fit by comparing the estimated histograms of selected DCT coefficients with those of the stego-image. Experimental results indicate that relative modifications as small as 10% of the usable DCT coefficients can be reliably detected. The method is tested on a diverse set of test images that include both raw and processed images in the JPEG and BMP formats.

1 Overview of Steganography and Steganalysis

Steganography is the art of invisible communication. Its purpose is to hide the very presence of communication by embedding messages into innocuous-looking cover objects. In today's digital world, invisible ink and paper have been replaced by much more versatile and practical covers for hiding messages – digital documents, images, video, and audio files. As long as an electronic document contains perceptually irrelevant or redundant information, it can be used as a "cover" for hiding secret messages. In this paper, we deal solely with covers that are digital images stored in the JPEG format.

Each steganographic communication system consists of an embedding algorithm and an extraction algorithm. To accommodate a secret message, the original image, also called the cover-image, is slightly modified by the embedding algorithm. As a result, the stego-image is obtained.

Steganalysis is the art of discovering hidden data in cover objects. As in cryptanalysis, we assume that the steganographic method is publicly known with the exception of a secret key. The method is secure if the stego-images do not contain any detectable artifacts due to message embedding. In other words, the set of stego-images should have the same statistical properties as the set of cover-images. If there exists an algorithm that can guess whether or not a given image contains a secret message with a success rate better than random guessing, the steganographic system is considered broken. For a more exact treatment of the concept of steganographic security, the reader is referred to [1–3].

The ability to detect secret messages in images is related to the message length. Obviously, the less information we embed into the cover-image, the smaller the probability of introducing detectable artifacts by the embedding process. Each steganographic method has an upper bound on the maximal safe message length (or the bit-rate expressed in bits per pixel or sample) that tells us how many bits can be safely embedded in a given image without introducing any statistically detectable artifacts. Determining this maximal safe bit-rate (or steganographic capacity) is a non-trivial task even for the simplest methods. Chandramouli et al. [4] give a theoretical analysis of the maximal safe bit-rate for LSB embedding in the spatial domain. Recently, Fridrich et al. [5,6] derived a more stringent estimate using dual statistics steganalysis.

The choice of cover-images is important because it significantly influences the design of the stego system and its security. Images with a low number of colors, computer art, images with a unique semantic content, such as fonts, should be avoided. Aura [7] recommends grayscale images as the best cover-images. He also recommends uncompressed scans of photographs or images obtained with a digital camera containing a high number of colors, and considers them safest for steganography.

The choice of the image format also makes a very big impact on the design of a secure steganographic system. Raw, uncompressed formats, such as BMP, provide the biggest space for secure steganography, but their obvious redundancy makes them very suspicious in the first place. Indeed, some researchers do not consider those formats for steganography claiming that exchanging uncompressed images is “equivalent” to using cryptography [8]. Never the less, most steganographic products available on the Internet work with uncompressed image formats or formats that compress data losslessly (BMP, PCX, GIF, PGM, and TIFF).

Fridrich et al. [9] have recently shown that cover-images stored in the JPEG format are a very poor choice for steganographic methods that work in the *spatial* domain. This is because the quantization introduced by JPEG compression can serve as a "semi-fragile watermark" or a unique fingerprint that can be used for detection of very small modifications of the cover-image by inspecting the compatibility of the stego-image with the JPEG format. Indeed, changes as small as flipping the least significant bit (LSB) of one pixel can be reliably detected. Consequently, one should avoid using decompressed JPEG images as covers for spatial steganographic methods, such as the LSB embedding or its variants.

Despite its proven insecurity, the method of choice of most publicly available steganographic tools is the LSB embedding. This paradigm can be adapted not only to raw formats but also to palette images after pre-sorting the palette (EZ Stego [10]) and to JPEG images (J-Steg [10], JP Hide&Seek [10], and OutGuess [11]).

Fridrich et al. [5,6] introduced the dual statistics steganalytic method for detection of LSB embedding in uncompressed formats. For high quality images taken with a digital camera or a scanner, the dual statistics steganalysis indicates that the safe bit-rate is less than 0.005 bits per sample, providing a surprisingly stringent upper bound on steganographic capacity of simple LSB embedding.

Pfitzmann and Westfeld [12] introduced a method based on statistical analysis of Pairs of Values (PoVs) that are exchanged during message embedding. For example, grayscales that differ in the LSBs only, could form these PoVs. This method, which became known as the χ^2 attack, is quite general and can be applied to many embedding paradigms besides the LSB embedding. It provides very reliable results when the message placement is known (e.g., for sequential embedding). Pfitzmann [12] and Provos [13] noted that the method could still be applied to randomly scattered messages by applying the same idea to smaller portions of the image while comparing the statistics with the one obtained from unrelated pairs of values. Unfortunately, no further details regarding this generalized χ^2 attack are provided in their papers, although Pfitzmann [12] reports that messages as small as one third of the total image capacity are detectable.

Farid [14] developed a universal blind detection scheme that can be applied to any steganographic scheme after proper training on databases of original and cover-images. He uses an optimal linear predictor for wavelet coefficients and calculates the first four moments of the distribution of the prediction error. Fisher linear discriminant statistical clustering is then used to find a threshold that separates stego-images from cover-images. Farid demonstrates the performance on J-Steg, both versions of OutGuess, EZ Stego, and LSB embedding. It appears that the selected statistics is rich enough to cover a very wide range of steganographic methods. However, the results are reported for a very limited image database of large, high-quality images, and it is not clear how the results will scale to more diverse databases. Also, the authors of this paper believe that methods that are targeted to a specific embedding paradigm will always have significantly better performance than blind methods.

Johnson and Jajodia [15] pointed out that some steganographic methods for palette images that preprocess the palette before embedding are very vulnerable. For example, S-Tools [10] or Stash [10] create clusters of close palette colors that can be swapped for each other to embed message bits. These programs decrease the color depth and then expand it to 256 by making small perturbations to the colors. This preprocessing, however, will create suspicious and easily detectable pairs (clusters) of close colors.

Recently, the JPEG format attracted the attention of researchers as the main steganographic format due to the following reasons: It is the most common format for storing images, JPEG images are very abundant on the Internet bulletin boards and public Internet sites, and they are almost solely used for storing natural images. Modern steganographic methods can also provide reasonable capacity without necessarily sacrificing security. Pfitzmann and Westfeld [16] proposed the F5 algorithm as an example of a secure but high capacity JPEG steganography. The authors presented the F5 algorithm as a challenge to the scientific community at the Fourth Information Hiding Workshop in Pittsburgh in 2001. This challenge stimulated the research presented in this paper.

In the next section, we give a description of the F5 algorithm as introduced in [16]. Then, in Sect. 3, we describe an attack on F5 and give a sample of experimental results. The limitations of the detection method and ways to overcome those limitations are discussed in Sect. 4. The paper is concluded in Sect. 5, where we also outline our future research.

2 The F5 Algorithm

The F5 steganographic algorithm was introduced by German researchers Pfitzmann and Westfeld in 2001 [16]. The goal of their research was to develop concepts and a practical embedding method for JPEG images that would provide high steganographic capacity without sacrificing security. Guided by their χ^2 attack, they challenged the paradigm of replacing bits of information in the cover-image with the secret message while proposing a different paradigm of incrementing image components to embed message bits. Instead of replacing the LSBs of quantized DCT coefficients with the message bits, the absolute value of the coefficient is decreased by one. The authors argue that this type of embedding cannot be detected using their χ^2 statistical attack.

The F5 algorithm embeds message bits into randomly-chosen DCT coefficients and employs matrix embedding that minimizes the necessary number of changes to embed a message of certain length. According to the description of the F5 algorithm, version 11, the program accepts five inputs:

- Quality factor of the stego-image Q;
- Input file (TIFF, BMP, JPEG, or GIF);
- Output file name;
- File containing the secret message;
- User password to be used as a seed for PRNG;
- Comment to be inserted in the header.

In the embedding process, the message length and the number of non-zero non-DC coefficients are used to determine the best matrix embedding that minimizes the number of modifications of the cover-image. Matrix embedding has three parameters (c, n, k), where c is the number of changes per group of n coefficients, and k is the number of embedded bits. In their paper [16], the authors describe a simple matrix embedding $(1, 2^k–1, k)$ using a “hash” function that outputs k bits when applied to $2^k–1$ coefficients.

The embedding process starts with deriving a seed for a PRNG from the user password and generating a random walk through the DCT coefficients of the cover-image. The PRNG is also used to encrypt the value k using a stream cipher and embed it in a regular manner together with the message length in the beginning of the message stream. The body of the message is embedded using matrix embedding, inserting k message bits into one group of $2^k–1$ coefficients by decrementing the absolute value of at most one coefficient from each group by one.

The embedding process consists of the following six steps:

1. Get the RGB representation of the input image.

2. Calculate the quantization table corresponding to quality factor Q and compress the image while storing the quantized DCT coefficients.
3. Compute the estimated capacity with no matrix embedding $C = h_{DCT} - h_{DCT}/64 - h(0) - h(1) + 0.49h(1)$, where h_{DCT} is the number of all DCT coefficients, $h(0)$ is the number of AC DCT coefficients equal to zero, $h(1)$ is the number of AC DCT coefficients with absolute value 1, $h_{DCT}/64$ is the number of DC coefficients, and $-h(1)+0.49h(1) = -0.51h(1)$ is the estimated loss due to shrinkage (see Step 5). The parameter C and the message length together determine the best matrix embedding.
4. The user-specified password is used to generate a seed for a PRNG that determines the random walk for embedding the message bits. The PRNG is also used to generate a pseudo-random bit-stream that is XOR-ed with the message to make it a randomized bit-stream. During the embedding, DC coefficients and coefficients equal to zero are skipped.
5. The message is divided into segments of k bits that are embedded into a group of 2^k-1 coefficients along the random walk. If the hash of that group does not match the message bits, the absolute value of one of the coefficients in the group is decreased by one to obtain a match. If the coefficient becomes zero, the event is called *shrinkage*, and the same k message bits are re-embedded in the next group of DCT coefficients (we note that LSB(d)= d mod 2, for $d > 0$, and LSB(d)=1– d mod 2, for $d < 0$).
6. If the message size fits the estimated capacity, the embedding proceeds, otherwise an error message showing the maximal possible length is displayed. There are rare cases when the capacity estimation is wrong due to a larger than anticipated shrinkage. In those cases, the program embeds as much as possible and displays a warning.

While the F5 algorithm does modify the histogram of DCT coefficients, the authors show that some crucial characteristics of the histogram are preserved, such as its monotonicity and monotonicity of increments. The F5 algorithm cannot be detected using the χ^2 attack because the embedding is not based on bit-replacement or exchanging any fixed Pairs of Values.

In the next section, we describe an attack on F5. It is based on the idea that one can accurately estimate the histogram of the cover-image from the stego-image. Because F5 modifies the histogram in a well-defined manner, we can calculate the number of modified coefficients by comparing the estimated histogram with the histogram of the stego-image.

3 Description of the Attack

We divided our attack on F5 into two separate parts: (1) Finding distinguishing statistical quantities T that correlate with the number of modified coefficients, and (2) Determining the baseline values of the statistics T. In fact, it is not that difficult to find a quantity that changes with embedded message length. For example, the number of coefficients equal to zero increases while the number of remaining non-zero coefficients decreases. Another measure that can be used is the “blockiness“ or the measure of discontinuity at the boundaries of the 8×8 grid. Actually, the blockiness is likely to increase for any method that embeds message bits by modifying the quantized DCT coefficients of the cover-JPEG image (for example, in [17,18] we use the blockiness increase as the distinguishing quantity to successfully attack the OutGuess [11]). What is difficult, however, is finding the baseline values or their estimates for the distinguishing statistics T – the original value(s) of T for the cover-image.

In the following subsection, we first analyze how F5 changes the histogram values. Then, we describe a method for obtaining the estimate of the cover-image histogram from the stego-image. We continue with a detailed description of a detection method that is capable of estimating the message length. Finally, we close Sect. 3 with experimental results and their discussion.

3.1 Analysis of Histogram Modifications

Let $h(d)$, $d = 0, 1, \ldots$ be the total number of AC coefficients in the cover-image with absolute value equal to d after the image has been compressed inside the F5 algorithm (Step 2 above). In a similar manner, we denote $h_{kl}(d)$ the total number of AC DCT coefficients corresponding to the frequency (k, l), $1 \leq k, l \leq 8$, whose absolute value is equal to d. The corresponding histogram values for the stego-image will be denoted using the capital letters H and H_{kl}.

Let us suppose that the F5 embedding process changes n AC coefficients. The probability that a non-zero AC coefficient will be modified is $\beta = n/P$, where P is the total number of non-zero AC coefficients ($P = h(1) + h(2) + \ldots$). Because the selection of the coefficients is random in F5, the expected values of the histograms H_{kl} of the stego-image are

$$\begin{aligned} &H_{kl}(d) = (1-\beta)h_{kl}(d) + \beta h_{kl}(d+1), && \text{for } d > 0, \\ &H_{kl}(0) = h_{kl}(0) + \beta h_{kl}(1), && \text{for } d = 0. \end{aligned} \qquad (1)$$

Let us further assume that we have an estimate $\hat{h}_{kl}(d)$ of the cover-image histogram (the baseline). We can use this estimate to calculate the expected values $H_{kl}(d)$ using Eq. (1) and estimate β as the value that gives us the best agreement with the cover-image histogram. We have experimented with different formulas for β and the best performance was obtained using the least square approximation. Because the first two values in the histogram (d=0 and d=1) experience the largest change during embedding (see Fig. 1), we calculate β as the value that minimizes the square error between the stego-image histogram H_{kl}, and the expected values $\hat{H}_{kl}(d)$ calculated from the estimated histogram $\hat{h}_{kl}$ using Eq. (1):

$$\beta_{kl} = \arg\min_{\beta} [H_{kl}(0) - \hat{h}_{kl}(0) - \beta\hat{h}_{kl}(1)]^2 + [H_{kl}(1) - (1-\beta)\hat{h}_{kl}(1) - \beta\hat{h}_{kl}(2)]^2. \quad (2)$$

The least square approximation in Eq. (2) leads to the following formula for β

$$\beta_{kl} = \frac{\hat{h}_{kl}(1)[H_{kl}(0) - \hat{h}_{kl}(0)] + [H_{kl}(1) - \hat{h}_{kl}(1)][(\hat{h}_{kl}(2) - \hat{h}_{kl}(1)]}{\hat{h}^2{}_{kl}(1) + [\hat{h}_{kl}(2) - \hat{h}_{kl}(1)]^2}. \quad (3)$$

The final value of the parameter β is calculated as an average over selected low-frequency DCT coefficients (k, l) $\in \{(1,2),(2,1),(2,2)\}$. We decided to not include the higher frequency coefficients due to problems with potential insufficient statistics especially for small images.

The reasons why we opted to work with histograms of individual low-frequency DCT coefficients rather than the global histogram will become apparent in Sect. 3.2 after we introduce the method for obtaining the cover-image histogram.

3.2 Estimating the Cover-Image Histogram

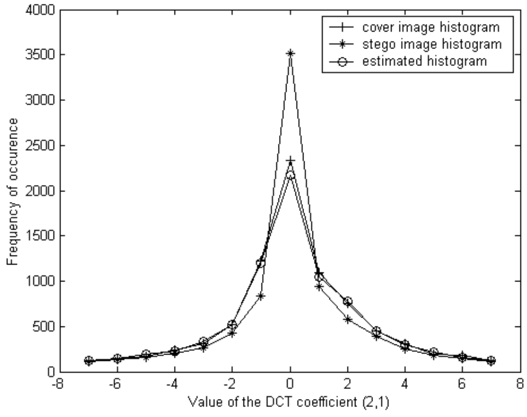

Fig. 1. The effect of F5 embedding on the histogram of the DCT coefficient (2,1)

Accurate estimation of the cover-image histogram h is absolutely crucial for our detection method to work. We first decompress the stego-image to the spatial domain, then crop the image by 4 columns, and recompress the cropped image using the same quantization matrix as that of the stego-image. The resulting DCT coefficients will provide the estimates $\hat{h}_{kl}(d)$ for our analysis. Because the accuracy of the estimates is the major factor influencing the detection accuracy, we include a simple preprocessing step to remove possible JPEG blocking artifacts from the cropped image before recompressing. We have experimented with several spatial blocking-removing algorithms, but the best results were obtained using a simple uniform blurring operation with a 3×3 kernel B, B_{22}=1−4e, $B_{21} = B_{23} = B_{12} = B_{32} = e$, and $B_{ij} = 0$ otherwise. This low-pass filter helps remove some spurious non-zero DCT coefficients produced by “discontinuities” at the block boundaries, which are in the middle of the 8×8 blocks of the cropped image.

According to our experiments, the estimated histogram is quite close to the histogram of the original image. We provide a simple heuristic explanation of why the method for obtaining the baseline histogram values is indeed plausible. In fact, unless the quality factor of the JPEG compression is too low (e.g., lower than 60), the stego-image produced by F5 is still very close to the cover-image both visually and using measures, such as the PSNR. The spatial shift by 4 pixels effectively breaks the structure of quantized DCT coefficients and subsequent low-pass filtering helps to reduce any spurious frequencies due to discontinuities at block boundaries. Thus, it is not surprising that the statistical properties of DCT coefficients are similar to those of the cover-image.

In Fig. 1, we show a typical example of how good the histogram estimate is when compared to the histogram of the original image. The graph shows the original histogram values $h_{21}(d)$ (crosses), histogram values after applying the F5 algorithm with maximal possible message, or $\beta = 0.5$ (stars), and the estimate of the original histogram (circles).

The main reason why we decided to use histograms of individual low-frequency DCT coefficients rather than the global image histogram is as follows. Even with the low-pass pre-filtering, the spatial shift by 4 pixels introduces some non-zero coefficients in high frequencies due to the discontinuities at block boundaries. And the values that are most influenced are 0, 1, and –1, which are the most influential in our calculations. Individual histograms of low frequency coefficients are much less susceptible to this onset of spurious non-zero DCTs.

We have identified two cases when the estimated histogram obtained using the algorithm described above does not give accurate values. This may occur, for example, when the cover-image sent to F5 has already been saved in the JPEG format with a different quality factor $Q_1 \neq Q$, or when the image contains some regular structure with a characteristic length comparable to the block size. Fortunately, both cases can be easily identified and our detection procedure correspondingly modified to obtain accurate results in those cases as well (see Sect. 4 and 5).

3.3 Estimating the True Message Length

Once the relative number of changes β has been estimated, we may attempt to further estimate the total message length. Let n be the total number of changes in quantized DCT coefficients introduced by the F5 algorithm. We can write n as $n = s + m$, where s is the shrinkage (modifications that did not lead to message bits embedding), and m is the number of changes due to actual message bit embedding. The probability of selecting a coefficient that may lead to shrinkage is $P_S = h(1)/P$. Since the coefficients are selected at random, the expected value of s is nP_S. Thus, we obtain the following formula:

$$m + nP_S = n,$$

which gives $m=n(1-P_S)$ for the number of changes due to message embedding. Assuming the $(1,2^k-1, k)$ matrix embedding, the expected number of bits per change $W(k)$ is

$$W(k) = \frac{2^k}{2^k - 1}k.$$

Thus, the unknown message length M can be calculated as

$$M = W(k)m = \frac{2^k}{2^k - 1} kn(1 - P_S) = \frac{2^k}{2^k - 1} k\beta P(1 - h(1)/P) = \frac{2^k}{2^k - 1} k\beta (P - h(1)),$$

where

$$P = \sum_{i \geq 0} h(i) \approx \sum_{i \geq 0} \sum_{\substack{k,l=1 \\ k+l>2}}^{8} \hat{h}_{kl}(i).$$

The parameter k can be derived from the knowledge of $n = \beta P$ and m and the estimated cover-image histogram by following the algorithm of determining the optimal matrix embedding as implemented in F5.

3.4 Experimental Results

We have created a database of 20 grayscale images with dimensions ranging from as small as 469×625 pixels up to 1336×1782 pixels. The images were obtained using ten different digital cameras and two scanners, resized to a smaller, randomly chosen size, and saved as BMPs. Then, we applied the F5 algorithm with quality factor 75 so that the ratio β of modified coefficients to the number of all non-zero, non-DC coefficients was 0, 0.25, and 0.5, corresponding to an empty message embedded, 25%, and 50% of usable coefficients modified. The estimated ratio β and its distribution across the test images are depicted in Fig. 2. All three Gaussian peaks are centered very close to the true value of β and all three are very well separated. In fact, based on this statistical data, the detection threshold $T = 0.125$ will lead to a false detection probability of 10^{-8}, probability of missing a message with $\beta = 0.25$ equal to 10^{-7}, and probability of missing a message with $\beta = 0.5$ (full capacity) equal to 10^{-32}.

To find out the limits of the detection methods, we have embedded a relatively short message of 4.5kB in 10 randomly chosen test grayscale BMP images (out of 20 images) all of the same dimensions 800×600. Table 1 shows the estimated ratio $\hat{\beta}$ and the estimated number of modifications $\hat{n}$ together with the actual values β and n.

4 Eliminating the Effects of Double Compression

When the cover-image is stored in the JPEG format, the F5 decompresses it first and then recompresses with a user-specified quality factor. After that, the message is embedded in the quantized DCT coefficients. This means that the stego-image has been double compressed before embedding. The double compression can have a profound effect on the image histogram and it complicates the detection.

The process of obtaining the baseline histogram from the cropped image as described in the previous section will produce a histogram similar to the broken line in Fig. 3 instead of the solid line from which the F5 started its embedding. Consequently, the estimated relative number of changes β may be quite different from the actual value. Fig. 4 shows the estimated β for a grayscale cover-image saved as JPEG with quality factors $Q_1 = 55$ to 95. Good accuracy is only obtained for values Q_1 close to the F5 quality factor of 75 and for high quality JPEGs with $Q_1 > 90$. The estimated β is particularly inaccurate when the quality factor of the stego-image Q_1 is lower than 75 (see the numbers in brackets in Table 3).

To address the problems with inaccurate detection when the cover-images are stored in the JPEG format, we proposed the following modification of our detection.

Table 1. The number of relative modifications of DCT coefficients $\beta = n/P$ and its estimate obtained using our detection method for 20 test images. Ten images contain a 4.5kB message, while the other 10 have only been compressed with F5. The absolute number of modified coefficients and its estimate are given in the last two columns

Img	β	$\hat{\beta}$	n	$\hat{n}$
1	0	0.106		11846
2	0.202	0.238	19845	21937
3	0	0.079		5214
4	0.259	0.273	20254	19490
5	0.244	0.265	21401	21011
6	0.234	0.276	20267	22040
7	0.216	0.248	19675	21176
8	0.347	0.409	24741	25873
9	0	0.044		2570
10	0	0.070		5124
11	0	0.103		6187
12	0.342	0.250	23589	15745
13	0.499	0.522	22775	21531
14	0	0.113		8386
15	0	0.078		4571
16	0.257	0.291	20164	20955
17	0	0.083		7222
18	0	0.073		4513
19	0.370	0.329	23930	19342
20	0.428	0.377	24278	19308

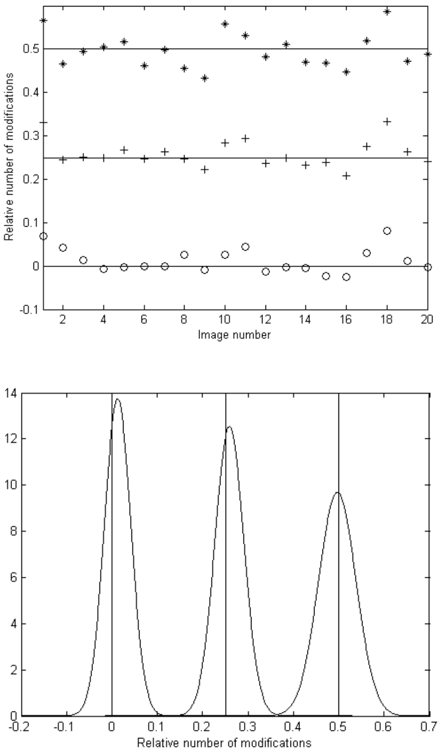

Fig. 2. Estimated number and distribution of relative modifications of DCT coefficients β in 20 test images. The lines correspond to the actual modifications with $\beta = 0, 0.25, 0.5$

We calculate the ratio β for a fixed set of quantization tables, $\{Q_1, Q_2, \ldots, Q_r\}$. For each quantization table, we run our detection scheme with one small modification – after cropping the decompressed filtered stego-image, we compress it with the quantization table Q_i and immediately decompress before proceeding with the rest of the baseline histogram estimation. Then, we calculate the estimated ratio β_i, $i = 1, \ldots, r$ in the usual manner. For each i and for each DCT mode kl, we calculate the L_2 distance $E^{(i)}{}_{kl}$ between the stego-image histogram H_{kl} and the histogram obtained using Eq. (1) with $\beta = \beta_i$:

$$E^{(i)}{}_{kl} = [H_{kl}(0) - \hat{h}_{kl}(0) - \beta_i \hat{h}_{kl}(1)]^2 + \sum\nolimits_j [H_{kl}(j) - (1-\beta_i)\hat{h}_{kl}(j) - \beta_i \hat{h}_{kl}(j+1)]^2 .$$

The final estimated ratio β is obtained as $\beta = \beta_t$, where $t = \arg\min_i \sum\nolimits_{kl} E_{kl}{}^{(i)}$, the sum being taken over all low-frequency coefficients that participate in our calculations (see Sect. 3.1).

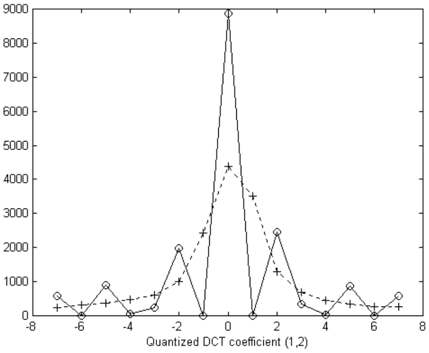

Fig. 3. Effect of double compression on the histogram of quantized DCT coefficients. The broken line is the image histogram with a single compression, the solid line after double compression with a lower quality factor being the first one. The histogram corresponds to the DCT coefficient (1,2)

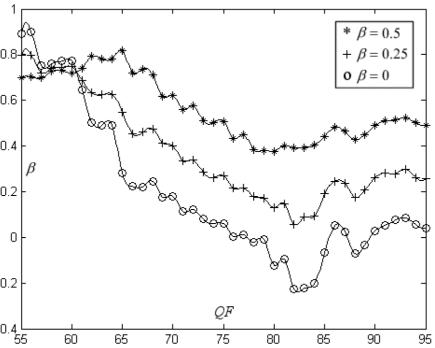

Fig. 4. Estimated number of relative modifications β for a grayscale 800×600 test cover-image saved as JPEG with quality factors QF = 55 to 95, and F5 quality factor 75, as a function of QF. Circles, crosses, and stars correspond to β = 0, 0.25, and 0.5, respectively

The estimated relative number of modifications improves dramatically when the double compression detection is added to the detection routine (see Table 2). The improvement in estimates due to incorporating double compression detection and correction is quite obvious. The table shows the estimated ratio β obtained without considering the effects of double compression (in brackets), and β calculated using the extended detection algorithm as described above. Although the overall accuracy of the estimated ratio β is somewhat lower when compared to the results obtained for cover-images that were not JPEG compressed, the results indicate that a reasonably accurate detection is still possible.

Table 2. Estimated β obtained with double compression correction and without (in brackets)

Image	Dimensions	$\beta = 0.00$	$\beta = 0.25$	$\beta = 0.50$
kangaroo.jpg	533×800	0.02 (–0.10)	0.26 (0.15)	0.47 (0.35)
portrait.jpg	469×625	–0.01 (0.14)	0.23 (0.48)	0.44 (0.79)
mcdonalds.jpg	960×1280	–0.02 (0.13)	0.24 (0.41)	0.50 (0.65)
kobe_pyramid.jpg	697×1045	0.02 (0.06)	0.28 (0.31)	0.53 (0.59)
bday.jpg	1050×1400	0.17 (–0.13)	0.37 (0.14)	0.56 (0.42)

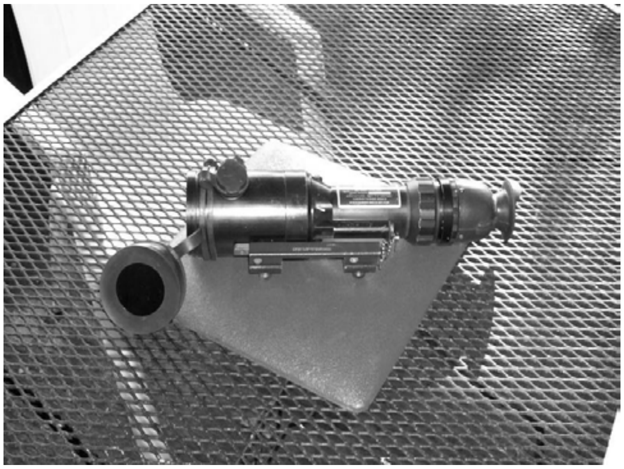

Fig. 5. Example of an image with spatial resonance. The same image cropped by 4 and 4 pixels has very different block frequency characteristics than the original image

Another case of test images that may produce large errors in our detection scheme are images that exhibit very different block frequency characteristics after the cropping. This "spatial resonance" may occur when the cover-image contains some regular structure with a characteristic length comparable to the block size, such as the metal grid in Fig. 5. Fortunately, it is easy to identify such images both visually and algorithmically and take appropriate measures. One possibility is to use those frequency modes that are most stable with respect to cropping and avoid those that exhibit strong resonant behavior. In our tests, we have encountered only two images with spatial resonance among hundreds of images randomly selected from different sources.

5 Conclusion

In this paper, we present an attack on the F5 steganographic algorithm as proposed by Pfitzmann and Westfeld in [16]. The attack is based on the idea that it is possible to estimate the cover-image histograms of individual low-frequency DCT modes by cropping the decompressed stego-image by 4 and 4 pixels and recompressing it again using the stego-image quantization matrix. After these baseline histograms are obtained, we determine the relative number of modified non-zero non-DC coefficients β as the value that minimizes the least square error between the stego-image histograms and the histograms obtained by embedding a message that leads to exactly β modifications. The detection algorithm estimates β, which can consequently be turned into an estimate of the secret embedded message.

When the cover-image is in some other format that the JPEG format, the detection results are very reliable and accurate. We demonstrated the performance of our detection method on a test database consisting of 20 grayscale images obtained with different digital cameras and scanners with various dimensions. The experimental results indicate that the detection threshold T = 0.125 leads to the probability of a false detection 10^{-8}, probability of missing a message with $\beta = 0.25$ equal to 10^{-7}, and probability of missing a message with $\beta = 0.5$ (full capacity) equal to 10^{-32}.

When the cover-images are stored in the JPEG format, the detection method must be modified to accommodate the effects of double JPEG compression produced by the embedding. The F5 always decompresses the cover-image and recompresses it using a user-defined quality factor. This leads to artifacts in coefficient histograms (jaggedness) that may introduce quite large detection errors. Fortunately, the previous JPEG compression can be estimated from the stego-image and the same compression/decompression that occurred prior to applying the F5 can be carried out for the cropped stego-image before deriving the estimated histograms for comparison. This small modification of the detection algorithm dramatically improves the performance and makes the accuracy and reliability of our results independent of the cover-image format.

The method for obtaining the cover-image histogram by cropping and low-pass filtering can in fact be used for designing detection mechanisms for other steganographic schemes that manipulate quantized DCT coefficients. We can use different statistical quantities rather than first-order statistics in the frequency domain to obtain their baseline values. For example, the increase of “blockiness” (the sum of spatial discontinuities at block boundaries) during embedding can be used as the distinguishing quantity for OutGuess [11]. Using this measure, we have been able to successfully attack OutGuess [17,18]. The blockiness measure increases with embedding for most steganographic schemes for JPEGs independently of their inner mechanisms. This opens up a new direction in steganalysis of JPEG images that yet needs to be further explored.

Acknowledgement

The work on this paper was supported by Air Force Research Laboratory, Air Force Material Command, USAF, under a research grant number F30602-00-1-0521. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation there on. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily

representing the official policies, either expressed or implied, of Air Force Research Laboratory, or the U. S. Government.

References

1. Anderson, R.J. and Petitcolas, F.A.P.: On the Limits of Steganography. IEEE Journal of Selected Areas in Communications: Special Issue on Copyright and Privacy Protection), Vol. 16(4) (1998) 474–481
2. Cachin, C.: An Information-Theoretic Model for Steganography. In: Aucsmith D. (eds.): Information Hiding: 2nd International Workshop. Lecture Notes in Computer Science, Vol. 1525. Springer-Verlag, Berlin Heidelberg New York (1998) 306–318
3. Katzenbeisser, S. and Petitcolas, F.A.P.: On Defining Security in Steganographic Systems. Proceedings of SPIE: Electronic Imaging 2002, Security and Watermarking of Multimedia Contents, Vol. 4675. San Jose, California (2002)
4. Chandramouli, R. and Memon, N.: Analysis of LSB Based Image Steganography Techniques. Proceedings of ICIP 2001 (CD version). Thessaloniki, Greece (2001)
5. Fridrich, J., Goljan, M., and Du, R.: Reliable Detection of LSB Steganography in Grayscale and Color Images. Proc. of ACM: Special Session on Multimedia Security and Watermarking. Ottawa, Canada (2001) 27–30
6. Fridrich, J., Goljan, M., and Du, R.: Detecting LSB Steganography in Color and Grayscale Images. Magazine of IEEE Multimedia: Special Issue on Security, Vol. Oct-Dec (2001) 22–28
7. Aura, T.: Practical Invisibility in Digital Communication. In: Anderson, R.J. (eds.): Information Hiding: 1st International Workshop. Lecture Notes in Computer Science, Vol.1174. Springer-Verlag, Berlin Heidelberg New York (1996) 265–278
8. Eggers, J.J., Bäuml, R., and Girod, B.: A Communications Approach to Image Steganography. Proceedings of SPIE: Electronic Imaging 2002, Security and Watermarking of Multimedia Contents, Vol. 4675. San Jose, California (2002)
9. Fridrich, J., Goljan, M., and Du, R.: Steganalysis Based on JPEG Compatibility. Proc. SPIE Multimedia Systems and Applications IV, Vol. 4518. Denver, Colorado (2001) 275–280
10. Steganography software for Windows, http://members.tripod.com/steganography/stego/software.html
11. Provos, N.: Defending Against Statistical Steganalysis. Proc. 10th USENIX Security Symposium. Washington, DC (2001)
12. Westfeld, A. and Pfitzmann, A.: Attacks on Steganographic Systems. In: Pfitzmann A. (eds.): 3rd International Workshop. Lecture Notes in Computer Science, Vol.1768. Springer-Verlag, Berlin Heidelberg New York (2000) 61–75
13. Provos, N. and Honeyman, P.: Detecting Steganographic Content on the Internet. CITI Technical Report 01-11. (2001)
14. Farid, H.: Detecting Steganographic Message in Digital Images. Technical Report, TR2001-412. Dartmouth College, New Hampshire (2001)
15. Johnson, N.F., Duric, Z., and Jajodia, S.: Information Hiding: Steganography and Watermarking - Attacks and Countermeasures. Kluwer Academic Publishers, Boston Dodrecht London (2000)
16. Westfeld, A.: High Capacity Despite Better Steganalysis (F5–A Steganographic Algorithm). In: Moskowitz, I.S. (eds.): Information Hiding. 4th International Workshop. Lecture Notes in Computer Science, Vol.2137. Springer-Verlag, Berlin Heidelberg New York (2001) 289–302
17. Fridrich, J., Goljan, M., and Hogea, D.: Attacking the OutGuess. Proc. ACM: Special Session on Multimedia Security and Watermarking, Juan-les-Pins, France (2002)

18. Fridrich, J., Goljan, M., and Hogea, D.: New Methodology for Breaking Steganographic Techniques for JPEGs. Submitted to SPIE: Electronic Imaging 2003, Security and Watermarking of Multimedia Contents. Santa Clara, California (2003)