Abstract

Steganography is the art of secret communication between two parties that not only conceals the contents of a message, but also its existence. Steganalysis attempts to detect the existence of embedded data in a steganographically altered cover file. Many algorithms have been proposed, but so far each has had some weakness that has allowed its effects to be detected, usually through first or second order statistical analysis of the image. In this paper, we propose a new approach to JPEG steganography that provides good stealthiness with global and dual histogram restoration. Our algorithm, named J4, is a new steganographic technique that improves on our previous system, J3. J4 uses dual histogram restoration along with matrix encoding to minimize detection by first as well as second order steganalysis. Existing blind steganalysis tests performed on J4 shows that it has a detection rate of 50-60% at 0.05-0.1 bit per non-zero coefficient (bpnz), as compared to 80-90% for F5, OutGuess, Steghide, MB1 and PQ. Since 50% detection rate is classified as random guess using a SVM classifier, J4 would be an ideal candidate for a steganographic algorithm.

1 Introduction

Steganography is a technique to hide data inside a cover medium in a way that the existence of any communication itself is undetectable as opposed to cryptography where the existence of secret communication is known but is indiscernible. Steganography has an edge over cryptography because it does not attract any public attention, and the data may be encrypted before being embedded in the cover medium. Hence, it incorporates cryptography with an added benefit of undetectable communication. Image files are the most common cover medium used for steganography. With resolution in most cases higher than human perception, data can be hidden in the “noisy” bits or pixels of the image file. Because of the noise, a slight change in the those bits is imperceptible to the human eye, although it might be detected using statistical methods (i.e., steganalysis).

Support Vector Machines (SVMs) have recently become popular to classify if a given image is stego or a cover [12]. The training data set consists of a number of features extracted from a set of cover and stego images. Based on this training data, SVM can build a prediction model that can classify the images. Steganalysis of JPEG images is based on statistical properties of the JPEG coefficients, since these statistical correlations are violated when DCT coefficients are modified during the embedding process. These statistical properties can be DCT features [5] or Markov features [17]. A more effective approach to steganalysis was achieved by combining, calibrating, and extending the DCT and Markov features together to produce a 274 merged feature set [13].
Results show that this method produces a better detection rate than using the DCT features or the Markov features alone. Our steganalysis experiment (discussed in section 7) uses this feature extractor with an SVM classifier.

The DCT features as proposed by Fridrich [5], are very effective in attacking any known steganographic system. Her algorithm is based on extraction of a number of DCT features from the given image set to detect embedded data using a classifier. J4 aims to compensate for some of those statistical changes after embedding data. In our previous work, J3 [11], we embed data by compensating for global histogram changes. We extend the idea of J3 into J4, which not only compensates for the global histogram but also for dual histograms, preserving the individual DCT modes. J4 conceals data in a way that it completely preserves the first order statistical properties [4] of the image and hence, is resistant to chi-square attacks [20] and most feature-based steganalysis techniques. In contrast to J3, we no longer embed any stop point information inside the header data. Instead, we estimate the embedding capacity of each coefficient pair and stop embedding any data once the estimated capacity is reached. The capacity estimation is done in such a way that there are enough coefficients left in each pair to compensate for any changes and restore the global as well as dual histogram after embedding.

In contrast to the popular approach, J4 embeds data in the zero coefficients for the lower coefficient indices. We consider zero coefficients since the number of zeros is extremely large as compared to other coefficients. Hence, to take advantage of the matrix embedding and minimize changes, we embed data in some zero coefficients. Note that only a small number of zero coefficients will be changed since the efficiency (bits embedded per coefficient change) increases due to a large number of available coefficients. Zero coefficients are changed in a way that doesn’t affect the histogram and the shape is retained. Matrix encoding, proposed by Crandall [3], can embed n bits of message in $2^n - 1$ cover bits by changing at most 1 bit. Hence, this encoding method is very useful when the message length is shorter than the maximum embedding capacity. F5 was the first steganography algorithm to use matrix encoding.

We compared our results with F5, nsF5 (no shrinkage F5 with wet paper codes) [8], Steghide, OutGuess [14], MB1 (model-based without blockiness) [15], MB2 (model-based with blockiness), PQ (Perturbed Quantization) [7], PQT (Texture based PQ) and PQE (Energy-based PQ). Based on 3000 sample JPEG images, our SVM-based steganalysis experiments show that J4 has a much lower detection rate than these algorithms except nsF5 which performs slightly better at 0.05 bits per non zero coefficient (bpnz).

The rest of the paper is organized as follows. In Section 2, we provide some background information on JPEG steganography and dual histograms. Section 3 deals with some of the previous work done in image steganography. In Section 4 and 5, we discuss our proposed J4 embedding and extraction algorithm in detail. Section 6 shows the performance of J4 in terms of ratio of 1’s to -1’s. Section 7 compares the steganalysis results of J4 with other popular algorithms mentioned previously. Finally, section 8 concludes the paper with reference to future work in this area.

2 Background

2.1 JPEG Steganography

There are two broad categories of image-based steganography that exist today: frequency domain and spatial domain steganography. The first digital image steganography was done in the spatial domain using LSB coding (replacing the least significant bit or bits with embedded data bits). Since JPEG transforms spatial data into the frequency domain where it then employs lossy compression, embedding data in the spatial domain before JPEG compression is likely to introduce too much noise and result in too many errors during decoding of the embedded data when it is returned to the spatial domain. These would be hard to correct using error correction coding. Hence, it was thought
that steganography would not be possible with JPEG images because of its lossy characteristics. However, JPEG encoding is divided into lossy and lossless stages. DCT transformation to the frequency domain and quantization stages are lossy, whereas entropy encoding of the quantized DCT coefficients (which we will call JPEG coefficients to distinguish them from the raw frequency domain coefficients) is lossless compression. Taking advantage of this, researchers have embedded data bits inside the JPEG coefficients before the entropy coding stage.

The most commonly used method to embed a bit is LSB embedding, where the least significant bit of a JPEG coefficient is modified in order to embed one bit of message. Once the required message bits have been embedded, the modified coefficients are re-encoded using entropy encoding to finally produce the JPEG stego image. During the extraction process, the JPEG file is entropy decoded to obtain the JPEG coefficients, from which the message bits are extracted from the LSB of each coefficient.

2.2 Dual Histogram and Individual DCT mode

In 2004, Fridrich et al. [5] proposed a steganalysis technique which was based on the DCT properties of the JPEG images. The DCT features extracted from the JPEG images outperformed existing techniques by a huge margin. Several of these features were based on the global histogram, dual histograms and the individual DCT modes. The individual DCT mode (also called individual histogram) is the occurrence of total number of each coefficient in a particular row and column of the 8x8 DCT matrix. For example, an individual histogram for (1,1) would be the occurrence of all coefficients at (1,1) location across all DCT blocks of the given JPEG images. Dual histograms are preserved automatically if an algorithm preserves all the individual histograms. In J4, we preserve all the individual histograms for the lower order coefficients in the lower indices of the DCT blocks. Let $d_k(i,j)$ denote the quantized DCT coefficient for $(i,j)$ coefficient in the $k$th block. The individual histogram for $(i,j)$ is defined as:

$$H^{i,j} = (h_x^{i,j}, ..., h_{-5}^{i,j})$$

$$(i, j) \in \{(2, 1), (3, 1), (1, 4), (1, 2), (2, 2), (3, 2), (1, 3), (2, 3), (1, 4)\}$$

where $h_x^{i,j}$ is the frequency of occurrence of coefficient $x$ at location $(i, j)$ in all the DCT blocks. The range $(-x,x)$ is set to $(-5, 5)$ since the frequency of higher order coefficients is too low to be considered for detection compared to the increased complexity.

3 Previous Work

Jsteg [19] was one of the first JPEG steganography algorithms. It was developed by Derek Upham, and embeds message bits in LSB of the JPEG coefficients. JP Hide&Seek [1] is another JPEG steganography program, improving stealth by using the Blowfish encryption algorithm to randomize the index for storing the message bits. This ensures that the changes are not concentrated in any particular portion of the image, a deficiency that made Jsteg more easily detectable. However, both of these algorithms are easily detected by the chi-square attack [20] since they equalize pairs of coefficients in a typical histogram of the image, giving a “staircase” appearance to the histogram.

F5 [21] is one of the most popular algorithms, and is undetectable using the chi-square technique. F5 uses matrix encoding along with permutating straddling to encode message bits. It also avoids making changes to any DC coefficients and coefficients with zero value. If the value of the message bit does not match the LSB of the coefficient, the coefficient’s value is always decremented, so that the overall shape of the histogram is retained. However, a one can change to a zero and hence the same message bit must be embedded in the subsequent coefficients until its value becomes non-zero, since zero coefficients are ignored on decoding. This technique also modifies the histogram of JPEG coefficients in a predictable manner. This is because of the shrinkage of ones converted to zeros
increases the number of zeros while decreasing the histogram of other coefficients and hence can be detected once an estimate of the original histogram is obtained [6].

Another popular algorithm is Steghide [10], which uses graph theory techniques to preserve the histogram. Two inter-changeable coefficients are connected by an edge in the graph with coefficients as vertices of the graph. The message is then embedded by swapping the two coefficients connected in the graph. Since the coefficients are swapped instead of replacing LSBs, it is difficult to detect any distortion using first order statistical analysis. But the steganalysis results show that it has a very high detection rate as compared to J4 even at lower embedding rates.

3.1 Statistical Restoration Techniques

Statistical restoration refers to the a class of embedding algorithms such that the first and/or higher order statistics are preserved after the embedding process. As mentioned earlier, embedding data in a JPEG image can lead to change in the typical statistics of the image which in turn can be detected by steganalysis. Most of the steganalysis methods existing today employ first and second order statistical properties of the image to detect any anomaly in the stego image. Statistical restoration is done to restore the statistics of the image as close as possible to the given cover image. Our algorithm, J4, falls under the category of first order statistical restoration or preservation schemes [14, 10, 18, 4, 9].

Solanki et. al [18] presented a statistical restoration technique where authors claim to achieve zero K-L divergence between the cover and the stego images while hiding at high rates. For JPEG images, they use 25% of the coefficients for hiding while preserving the rest for compensation. This approach is not very efficient because it does not use all the potential coefficients for hiding data. The coefficients in the compensation stream are modified using minimum mean-squared error criteria.

Another higher order statistical restoration technique has been presented by Sarkar et al. [16] where they use the earth-mover’s distance (EMD) technique to restore the second order statistics. EMD is a popular distance metric used in computer vision application. However, the authors have only considered the horizontal transitions probability in both inter/intra block dependency. They have not considered the diagonal and the vertical transitions which are also an important factor to restore the second order statistics. Moreover, they do not provide any concrete steganalysis results to prove the effectiveness of their method.

4 J4 Embedding Algorithm

As discussed before, J4 tries to restore the individual histograms as well as the global histogram. Restoration of the individual histograms would also result in the restoration of the global histogram. However, J4 doesn’t restore the individual and global histogram of 1, -1, and 0 coefficients. This
is to reduce the overall number of changes. To minimize impact, the ratio of 1’s to -1’s is always maintained throughout the embedding process. The embedding process of J4 goes through a number of steps as discussed below. Figure 4 shows the embedding module of J4.

4.1 Pre-processing Stage:

Preprocessing estimates embedding capacity for each coefficient pair. In order to simplify the algorithm, the DCT coefficients of an 8x8 matrix are considered as a one dimensional array where the coefficients are arranged in zigzag order. The right half of the coefficients in the array are high frequency components, and are mostly zeros after quantization. Therefore, we only consider the first 29 elements of the array for embedding and compensation since these hold most of the non-zero coefficients. The first element, that is, the DC coefficient is ignored. The frequency count of each coefficient is calculated for each index position in the array over all the DCT blocks. This arrangement can be viewed as a two dimensional bin or a matrix, where the row signifies the index in the DCT array and the column signifies the coefficient value. Hist\(_i\)(y) represents the total occurrences of coefficient value \(y\) at position \(i\) in each DCT array of a JPEG image. Once the individual histograms are calculated, capacity of each of the pairs in the bins is estimated theoretically (omitted due to page limitations). For example, we estimate the capacity of pair \((2y, 2y + 1)\) for each position \(i\) in the array, also represented as Est\(_i\)(2\(y\), 2\(y\) + 1). The estimated capacity is kept in separate 2-D bins. We also have an additional 2-D bin that keeps track of the current number of coefficients that have been used from a given pair, denoted by Used\(_i\)(2\(y\), 2\(y\) + 1), where \((2y, 2y + 1)\) form a coefficient pair and \(i\) is the coefficient index.

4.2 Header Information:

Once the preprocessing is over, we determine the appropriate values of \(n\) and \(k\) for matrix encoding. The matrix encoding is denoted by \((1, n, k)\), which means \(k\) bits of message can be embedded in \(n\) coefficients by changing at most 1 coefficient out of \(n\), where \(n = 2^k - 1\). The ratio \(n/k\) can be obtained by summing the total estimated capacity in bits divided by the total message bits. The receiver needs to know \(k\) in order to properly decode the message, so it is stored in the header bits of the message. The header also stores useful information such as the data length, coefficient threshold, and DCT array index limit. The header information itself is not matrix embedded since the receiver needs to decode the header bits to know the embedding rate, \(k\). The structure of the header is given in Table 4.2.

<table>
<thead>
<tr>
<th>4 Bits</th>
<th>20 Bits</th>
<th>4 Bits</th>
<th>2 Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value of matrix encoding, (k)</td>
<td>Data length in bytes, (M_L)</td>
<td>Coefficient threshold, (Coeff_{th})</td>
<td>DCT index threshold, (Index_{th})</td>
</tr>
</tbody>
</table>

Table 1: Header structure for J4 algorithm

**Explanation of Header fields:**

- \(k\) = Value of \(k\) in matrix encoding \((1, 2^k - 1, k)\).
- \(M_L\) = total message length in bytes, not including the length of header.
- \(Coeff_{th}\) = absolute value of upper coefficient limit for DCT coefficients. Any DCT coefficient value higher than this will be ignored during the embedding process.
- \(Index_{th}\) = represents the boundary index in the 1-D 8 x 8 zigzag array after which we ignore all the coefficients. The actual index position is encoded using two bits, where 00 = 28, 01 = 36, 10 = 43, 11 = 54. J4 by default uses value 00, i.e. 28 coefficients in the array for embedding, since half of the values in the JPEG coefficients are zero in a typical image.
4.3 Embedding Stage:

The cover image is first entropy decoded to obtain the JPEG coefficients. A pseudo-random number generator is used to visit the coefficients in random order to embed the encrypted message, seeded using shared password $P$. The algorithm always makes changes to the coefficients in a pairwise fashion; e.g. $\langle 2, 3 \rangle$ corresponds to a pair. A JPEG coefficient with a value of 2 will only change to a 3 to encode message bit 1, and a coefficient with a value 3 will only change to 2 to encode message bit 0.

In contrast to the general trend of not changing any zero valued coefficients, J4 makes changes to the zero coefficients. This is done in order to leverage the potential of an extremely high number of zero coefficients in a typical JPEG image. Since we use matrix encoding, using the zero coefficients results in an overall reduction in the number of coefficient changes and hence reduces detection. This heuristic is proven by the extremely low detection rate discussed in the steganalysis section. Coefficients with value 0, 1 and -1 have a different embedding strategy since their frequency is very high as compared to other coefficients, and form a triplet, $\langle -1, 0, 1 \rangle$. A -1 or 1 coefficient encodes a message bit 1, while 0 encodes message bit 0. To encode message bit 0 in a coefficient with value 1 or -1, we always change its value to 0. Similarly, to encode bit 1 in 0 coefficient, we change it to either a 1 or a -1. Change of a 0 to a 1 or -1 would depend on the total number of changes made to 1 and -1 before. The changes are done in such a way that the approximate ratio of -1 and 1 is maintained as in the cover file to thwart chi-square attacks. The algorithm keeps track of the imbalance in 1 and -1. If the imbalance in -1 exceeds that of 1, a zero would be changed to a -1 instead of 1 and vice-versa. Hence, the overall shape of the histogram is retained as in the cover image.

The embedding coefficient pairs are $\langle -2n, -2n-1 \rangle \cdots \langle -2, -3 \rangle, \langle -1, 0, 1 \rangle, \langle 2, 3 \rangle \cdots \langle 2n, 2n+1 \rangle$, where $2n+1$ and $-2n-1$ are the threshold limits for positive and negative coefficients, respectively, i.e. $|\langle -2n, -2n-1 \rangle| = |\langle 2n, 2n+1 \rangle| = \text{Coeff}_{th}$. During the embedding process, if the number of bits encoded for a particular pair, $(x, y)^i$ equals the estimated value, $\text{Est}^i(x, y)$, we stop considering pair $(x, y)^i$ for matrix embedding. Here $(x, y)^i$ denotes coefficients $x, y$ at index $i$ in the 1-D DCT array. Unused coefficients for that pair will be used later to compensate for the imbalance. Estimation ensures that enough coefficients are left untouched in order to restore the global and dual histogram after the embedding process. Note that full coefficient restoration is crucial as the receiver must calculate the same estimated capacity in order to decode the data properly and to know when to stop. Since the estimated capacity depends on the total number of each coefficient, we need to restore each coefficient pair for each index fully. If we cannot restore the coefficient fully, we need to make sure that estimation of the coefficient after the compensation would yield the same value as before the embedding process.

5 J4 Extraction Module

The extraction algorithm is simple, as the receiver has only to calculate the estimated capacity of each pair using $k$ from header and stop decoding that pair when the estimated capacity is equal to the number of bits used up in that pair. Password $P$ is used to generate the random number sequence used to permute the coefficient indices for visitation order. The header is decoded first to get the values of $k$, message length, coefficient threshold, and the index threshold.

Once all the header bits have been extracted, the extraction process starts decoding the message bits, taking care to stop extraction from a coefficient pair once its estimated capacity has been reached for each index.
6 Results

The algorithm was implemented in Java which includes code to 1) decode a JPEG image to get the JPEG coefficients, 2) embed data in eligible coefficients, 3) balance the dual histograms to their original values, and finally, 4) re-encode the image in JPEG format with modified coefficients while preserving the original quantization tables and other properties of the image. Tests were performed on 3000 different JPEG color images of varying size and texture obtained from National Geographic. Every image was embedded with random data bits using a randomly generated password.

6.1 Increase in 1, -1, and 0 Coefficients

J4 doesn’t compensate for changes made to -1, 1, and 0 coefficients. However, as mentioned earlier, the ratio of the total number of 1 and -1 coefficients is maintained in order to retain the shape of the histogram and reduce detection. Also, J4 doesn’t make changes to any 0, 1, or -1 coefficients outside the $Index_{th}$ range. This is done to ensure we don’t change a 0 to a 1 in the right half of the 1-D DCT array where most of the coefficients are 0. Figure 2 shows the percentage increase in total number of 1 and -1 coefficients for 0.05 bits per non zero coefficient. The figure shows that the increase in 1 and -1 coefficients doesn’t exceed more than 2% of its initial count. It also shows that the increase is almost identical in both since the lines overlap in most of the areas. This confirms that the ratio of 1 to -1 coefficients is maintained even after the embedding process. The secondary axis shows the decrease in the number of zeros. The total number of zeros will decrease since the number of zeros is much more than 1 and -1 and hence, more zeros will change to either 1 and -1 as compared to 1 and -1 being changed to zeros. Again, the percentage decrease in number of zeros is below 0.2 percent, which is negligible. Around 3000 images were used for this experiment, but we have only shown a random subset of them for simplicity.

![Figure 2: Percentage increase and decrease in 1, -1 and 0 for 0.05 $bpmz$](image)

7 Steganalysis of J4

Steganalysis experiments for J4 are based on Support Vector Machines (SVM) for classification of images embedded with the following stego algorithms: F5, nsF5, Outguess, Steghide, MB1, MB2, PQ, PQT, PQE and J4 along with the cover images. We use soft margin SVM (C-SVM) with RBF (Radial Basis Function) kernel, which is one of the most popular choices of kernel type for SVMs.
We use “LIBSVM” [2] tool, which is a library for SVM classification. The best values of \((C, \gamma)\) were used by doing cross validation (grid search) using the tool provided in the LIBSVM library. The experiments use a feature extractor that extracts 274 merged Markov and DCT features for steganalysis as mentioned in [13]. We used this merged feature extractor since it is state of the art, and outperforms DCT or Markov based steganalysis by itself, as shown in the authors’ results.

3000 JPEG color images with different texture and size ranging from 60 KB to 1000 KB were used for the steganalysis experiment. Every image was embedded with random data using the 10 above mentioned algorithms. At the end of embedding process, we have 10 sets of images containing 3000 stego images in each set. Each set consists of all the stego images embedded with only one of the 10 algorithms. We also have one set of cover images without any embedding. 70% of the images from each set were used for training and the remaining 30% were used for testing. The training and testing sets were mutually exclusive. We performed 100 iterations of each experiment by randomizing the training and testing data to get more accurate results.

### 7.1 Binary classification

We first performed a binary classification where only one of the stego sets and the cover set were used for training and testing. We performed this binary classification for all the 10 algorithms. The classification experiment was performed with 0.05, 0.1, 0.2, and 0.3 bits per non-zero coefficients (\(b_{pnz}\)). The results are shown in Table 7.1.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>(b_{pnz} = 0.05)</th>
<th>(b_{pnz} = 0.1)</th>
<th>(b_{pnz} = 0.2)</th>
<th>(b_{pnz} = 0.3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>J4</td>
<td>47.10 51.55</td>
<td>58.28 57.08</td>
<td>76.47 77.83</td>
<td>88.99 87.95</td>
</tr>
<tr>
<td>F5</td>
<td>93.44 92.06</td>
<td>93.61 92.86</td>
<td>96.94 96.93</td>
<td>98.50 98.30</td>
</tr>
<tr>
<td>Outguess</td>
<td>97.84 97.45</td>
<td>97.79 97.69</td>
<td>97.73 97.51</td>
<td>97.65 97.64</td>
</tr>
<tr>
<td>Steghide</td>
<td>83.26 86.41</td>
<td>93.66 95.27</td>
<td>97.24 97.64</td>
<td>98.45 98.09</td>
</tr>
<tr>
<td>nsF5</td>
<td>47.43 48.21</td>
<td>59.19 60.40</td>
<td>83.77 85.48</td>
<td>92.80 93.78</td>
</tr>
<tr>
<td>MB1</td>
<td>78.13 76.65</td>
<td>92.96 94.87</td>
<td>97.34 98.18</td>
<td>98.52 98.95</td>
</tr>
<tr>
<td>MB2</td>
<td>80.74 82.43</td>
<td>93.59 96.16</td>
<td>97.49 98.37</td>
<td>NA NA</td>
</tr>
<tr>
<td>PQ</td>
<td>97.11 98.78</td>
<td>97.18 98.86</td>
<td>98.32 99.05</td>
<td>98.76 99.21</td>
</tr>
<tr>
<td>PQT</td>
<td>97.67 98.11</td>
<td>97.77 98.84</td>
<td>98.23 99.12</td>
<td>99.78 99.01</td>
</tr>
<tr>
<td>PQE</td>
<td>98.15 98.45</td>
<td>98.95 98.73</td>
<td>98.55 99.54</td>
<td>99.14 98.97</td>
</tr>
</tbody>
</table>

Table 2: Comparison of detection rate (in %) of J4 with other algorithms using SVM binary-classifier with 0.05, 0.1, 0.2, and 0.3 \(b_{pnz}\). TP = True Positive, TN = True Negative.

Results in Table 7.1 show that J4 outperforms other algorithms (with the exception of \(nsF5\)) by a huge margin in terms of detection rate with \(b_{pnz}\) of 0.05. \(nsF5\) doesn’t do any compensation. J4 incurs extra cost by compensating for all the individual DCT modes. All the dual histograms are exactly the same as the cover image in J4. “True Positive” (TP) here refers to the correct prediction accuracy for the stego images whereas “True Negative” (TN) refers to the correct prediction accuracy of cover images against that particular stego algorithm. Hence, the lower the TP and TN are, the better the stego algorithm is. Results show that the SVM classifier was only able to classify 47% of the images in J4 category at 0.05 \(b_{pnz}\). It classified 51% of the J4 images as cover images, which proves that J4 resembles the characteristics of a cover image when the payload is less. Other algorithms classification rate was more than 80% on average at the same embedding rate. With 0.1 \(b_{pnz}\), J4 has a true positive rate of 58% whereas most algorithms have a true positive rate of more than 90% except \(nsF5\). It also outperforms \(nsF5\) by a small margin at 0.1, 0.2, and 0.3 \(b_{pnz}\). Since 50% detection rate is classified as a random guess, a detection rate of 50-60% for J4 proves that J4 could be an ideal candidate for a JPEG steganography algorithm.
8 Conclusion

J4 is a new JPEG steganography algorithm that uses LSB encoding to embed data and individual and global histogram compensation to balance all the coefficients changed during the embedding process. J4 makes changes to the coefficients in a way that the individual histograms are preserved as in the cover image. The preservation scheme does not apply to 1, 0, and -1 coefficients. This is done to leverage the high number of these coefficients for increased efficiency since J4 uses matrix embedding to decrease the number of coefficient changes. All the coefficients except 1, 0, and -1 are changed in pairs. Theoretical estimation ensures that enough coefficients are left after the embedding process to compensate for the changes to the coefficients at each index position in the DCT array.

We compared J4 with 9 popular algorithms: F5, nsF5, Steghide, OutGuess, MB1, MB2, PQ, PQT and PQE. Extensive steganalysis performed on these algorithms using state of the art methods reveal a detection rate for J4 at around 47%, 58%, 76% and 88% at 0.05, 0.1, 0.2 and 0.3 bits per non-zero coefficients (bpnz), respectively. On the other hand, the average detection rate of other algorithms except nsF5 were 90%, 95%, 97% and 98% for the same bpnz. The detection rate for other algorithms were around 40% higher on average compared to J4. nsF5 performs slightly better than J4 at 0.05 bpnz, but J4 outperforms nsF5 with a small margin at higher embedding rates. Moreover, nsF5 doesn’t do any compensation in contrast to J4. Although the steganalysis method (feature extractor) in the experiments uses both first and second order statistics to detect anomalies and J4 is a first-order restoration scheme, it is still able to perform extremely well on lower data rates and beat this steganalysis system. Since 50-60% detection is classified as random guess in SVM classification, the results prove that J4 would be an ideal candidate for embedding data at low rates.

References


